Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

Dan Bleichman
CogSim Technologies

Rob Langlois
R.G. Langlois Engineering Analysis

Michael Lichodzijewski and Dave Brennan
Martec Limited

Prepared by:
Martec Limited
Halifax, Nova Scotia

Contract Project Manager: Dave Brennan
Contract Number: W7707-063647/001/HAL
Contract Scientific Authority: Kevin McTaggart

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2011-261
October 2011

Canada

This page intentionally left blank.

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

Dan Bleichman
CogSim Technologies

Rob Langlois
R.G. Langlois Engineering Analysis

Michael Lichodzijewski
Martec Limited

Dave Brennan
Martec Limited

Prepared by:

Martec Limited
Halifax, Nova Scotia

Project Manager: Dave Brennan
Contract Number: W7707-063647/001/HAL
Contract Scientific Authority: Kevin McTaggart

Defence R&D Canada – Atlantic
Contract Report
DRDC Atlantic CR 2011-261
October 2011

Approved by

Neil Pegg
Head/Warship Performance

Approved for release by

Calvin Hyatt
Chair/Document Review Panel

Abstract

In order to carry out the Navy's mission effectively, fleet units must be capable of remaining at sea for prolonged periods of time, possibly in areas of the world where friendly re-supply ports are not available, and remain fully ready to carry out any assigned tasks. Supply ships are equipped to replenish combatants underway with fuel, ammunition, provisions, and spare parts.

The Canadian Department of National Defence is looking to use a simulation infrastructure called the High Level Architecture (HLA) to provide integrated, joint simulation environments for the development of tactics and doctrine as well as for training and systems acquisition. In order to support this effort, the Warship Performance Section at DRDC Atlantic has initiated a research and development effort aimed at producing a simulation of ship replenishment at sea.

The objective of the proposed technical solution described in this report is to provide a simulation environment that models the interactions between the various components in order to simulate conditions that lead to the adverse events of payload immersion and RAS gear breakage.

Résumé

Afin de remplir efficacement la mission de la Marine, les unités de la flotte doivent pouvoir demeurer en mer pendant de longues périodes de temps, possiblement dans des régions du monde où des ports de ravitaillement amis ne sont pas disponibles, tout en demeurant entièrement prêts à effectuer toutes tâches attribuées. Les navires ravitailleurs sont équipés pour réapprovisionner les combattants faisant route en carburant, munitions, provisions et pièces de rechange.

Le Ministère de la Défense nationale du Canada désire utiliser une infrastructure de simulation appelée architecture de haut niveau (HLA) afin de fournir des environnements de simulation interarmées intégrés en vue de l'élaboration de tactiques et de doctrine ainsi que pour l'entraînement et l'acquisition de systèmes. Afin d'appuyer cet effort, la Section de l'évaluation de la performance des navires de guerre de RDDC Atlantique a lancé un projet de recherche et développement en vue de produire la simulation de ravitaillements de navires en mer.

L'objectif de la solution technique proposée décrite dans le présent rapport est de fournir un environnement de simulation qui modélise l'interactions entre les divers composants afin de simuler les conditions qui mènent aux événements indésirables comme l'immersion de la charge utile et le bris du matériel de REM.

This page intentionally left blank.

Executive summary

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

Dan Bleichman, Rob Langlois, Michael Lichodzijewski, Dave Brennan;
DRDC Atlantic CR 2011-261; Defence R&D Canada – Atlantic; October 2011.

Background: The High Level Architecture (HLA) is widely used for developing distributed simulations of complex systems. In recent years, HLA has been used to develop simulations of naval platform systems. The application of HLA to naval platform systems introduces new challenges for running high fidelity physics-based models as distributed simulations.

Principal results: This report describes an HLA federation for simulation of ship replenishment at sea (RAS). The federation uses ship motion, seaway, and visualization simulation components that were previously developed at DRDC Atlantic. A new replenishment gear simulation model was developed to determine cable tensions and payload location during replenishment operations.

Significance of results: The RAS simulation demonstrates the successful implementation of an HLA federation using both existing and new physics-based models. The RAS simulation can be used to examine ship motions, cable tensions, and payload location during replenishment at sea. The implemented federation does not currently model hydrodynamic interaction forces between ships or collision forces between the payload and ships.

Future work: An HLA federate will be developed for modelling hydrodynamic interaction forces. Future efforts will also examine modelling of collision forces between payloads and ships.

Sommaire

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

Dan Bleichman, Rob Langlois, Michael Lichodzijewski, Dave Brennan ;
DRDC Atlantic CR 2011-261 ; R & D pour la défense Canada – Atlantique ;
octobre 2011.

Introduction : L'architecture de haut niveau (HLA) est très utilisée pour l'élaboration de simulations réparties de systèmes complexes. Au cours des dernières années, la HLA a servi à l'élaboration de simulations de systèmes de plateforme navale. L'application de la HLA aux systèmes de plateforme navale présente de nouveaux défis pour l'exécution de modèles très fiables et fondés sur des éléments physiques en tant que simulations réparties.

Résultats : Le présent rapport décrit une fédération HLA pour la simulation de ravitaillement en mer (REM). La fédération utilise des composants de simulation de mouvement de navire, de voie maritime et de visualisation déjà développés chez RDDC Atlantique. Un nouveau modèle de simulation de ravitaillement a été développé afin de déterminer les tensions des câbles et les emplacements de charge utile pendant les opérations de ravitaillement.

Portée : La simulation REM a démontré la mise en œuvre réussie d'une fédération HLA en utilisant à la fois les modèles existants et les nouveaux modèles fondés sur des éléments physiques. La simulation REM peut servir à étudier les mouvements de navire, les tensions des câbles et les emplacements de charge pendant les opérations de ravitaillement en mer. La fédération mise en œuvre ne modélise pas actuellement les forces d'interaction hydrodynamique entre les navires ni les forces de collision entre la charge utile et les navires.

Recherches futures : Un fédéré HLA sera développé pour la modélisation des forces d'interaction hydrodynamique. Les prochaines étapes permettront de se pencher sur la modélisation des forces de collision entre les charges utiles et les navires.

Table of contents

List of figures

List of tables

1 Introduction

1.1 Background

In order to carry out the Navy's mission effectively, fleet units must be capable of remaining at sea for prolonged periods of time, possibly in areas of the world where friendly re-supply ports are not available, and remain fully ready to carry out any assigned tasks. Supply ships are equipped to replenish combatants underway with fuel, ammunition, provisions, and spare parts [1].

The Canadian Department of National Defence is looking to use a simulation infrastructure called the High Level Architecture (HLA) to provide integrated, joint simulation environments for the development of tactics and doctrine as well as for training and systems acquisition [2]. In order to support this effort, the Warship Performance Section at DRDC Atlantic has initiated a research and development effort aimed at producing a simulation of ship replenishment at sea.

1.2 Problem Statement

In order to simulate the conditions leading to these adverse events, the interaction of ships, RAS gear, and environmental conditions must be modelled. Current methods of simulation either limit the interaction of these components or do not include the simulation of specific elements.

1.3 Project Objectives

The objective of the proposed technical solution is to provide a simulation environment that models the interactions between the various components in order to simulate conditions that lead to the adverse events of payload immersion and RAS gear breakage.

1.4 Technical Approach

A prototype of the HLA RAS federation was developed in an initial phase of the project. The prototype of the HLA RAS federation consisted of 4 federates [3]. In this final version of the HLA RAS federation the number of federates incorporated into the federation was increased to 10, where each federate can be replaced/upgraded independently of the other federates. This architecture allows for greater flexibility and a gradual development where federates can be added or replaced by ones of a higher fidelity as development progresses. The only constraints on development of specific federates is that they have to comply with a predefined federation object model (FOM) and a few other federation agreements.

Most federates consist of an HLA front-end and a mathematical modelling back-end where the mathematical modelling end can be upgraded over time (for better fidelity) while maintaining present external interfaces (the HLA front-end). The following components were modelled:

1. Execution manager federate
2. RAS gear federate
3. Supply-ship motion federate
4. Receiving-ship motion federate
5. Supply-ship helm federate
6. Receiving-ship helm federate
7. Seaway federate
8. Data logger federate
9. Hydrodynamic interaction forces federate
10. Visualizer federate

While each one of the above are modelled as separate HLA federates, the seaway federate has relatively modest capabilities at present, and serves mainly as a placeholder for future development.

1.5 Report Contents

In the report that follows, a detailed review of the work that was carried out is provided. This review includes a description of the RAS federation implementation. In addition, the results generated by running the federation for a series of verification test cases are also documented.

2 RAS Federation Implementation

The RAS gear simulation was implemented as a High Level Architecture (HLA) federation in order to satisfy the requirement for modularity of simulation components so that specific models (such as the RAS gear model or the ship motion model) would be easily replaceable by higher fidelity components when available or by different implementations of a given model.

2.1 Introduction

While the overall objective of this project was to provide a simulation environment that was able to model the behaviour of RAS gear, an important secondary objective was the provision of a flexible, extendable HLA federation that could form a strong basis for future development. The performance of the HLA RAS federation was confirmed while using high fidelity models of both the RAS gear equipment and the ship motion.

The RAS federation is a time-stepped HLA federation that consists of 10 federates. This federation uses a customized Federation Object Model (FOM) that is based upon the Real-Time Platform Reference (RPR) FOM Version 2.0 draft 17 [4]. Each one of the 10 federates is an independent application (linked against third party HLA/RTI libraries) that represents a specific model (or a set of models) and can be replaced by a different implementation as long as the new implementation complies with the Simulation Object Model (SOM) and the FOM.

Due to the distributed nature of HLA, a federate can share hardware or be distributed over a network of PCs where each PC can host either one or all of the federates (see Figure 1). In its current configuration, the RAS simulation is not computationally intensive, and as a result, a single PC (at 2 GHz CPU speed) can handle the full federation (excluding the visualizer federate) at near real-time performance. The HLA architecture was chosen for the RAS simulation since HLA at present is the de facto standard for distributed simulations and provides sound time management functionality to support a time stepped simulation. At present, the RAS federation consists of the following federates:

1. Execution manager federate
2. RAS gear federate
3. Supply-ship motion federate
4. Receiving-ship motion federate
5. Supply-ship helm federate

6. Receiving-ship helm federate
7. Seaway federate
8. Data logger federate
9. Hydrodynamic interaction forces federate
10. Visualizer federate

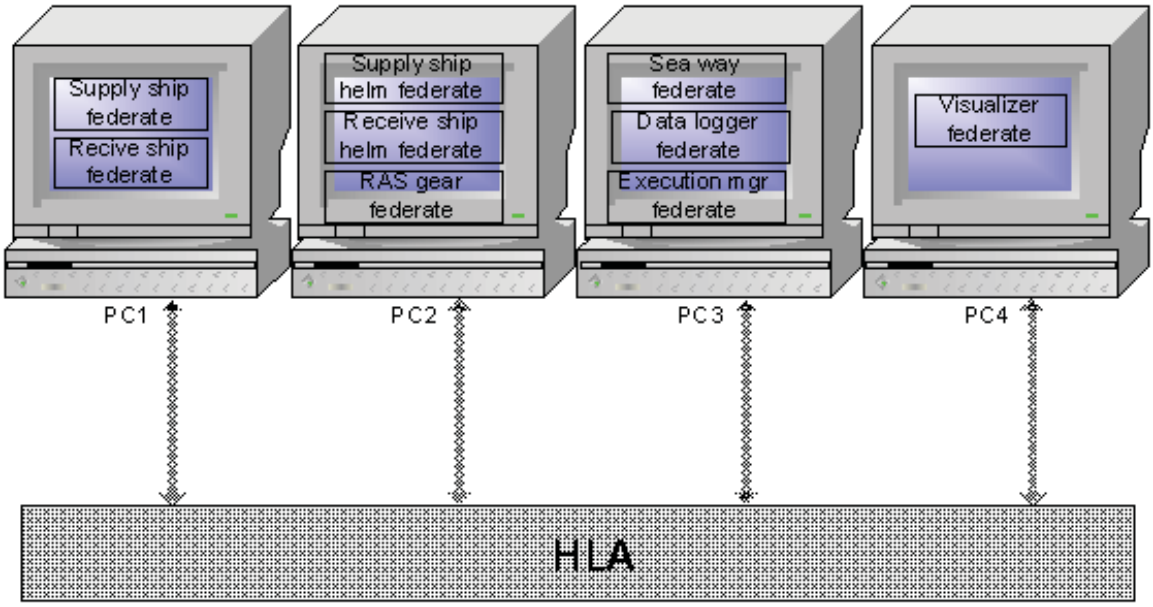

Figure 1: *RAS federation.*

Section 2.2 provides the reader with a basic introduction to HLA and the Run-Time Infrastructure (RTI), while Section 2.3 provides a discussion of HLA time management. Section 2.4 provides additional detail related to the various federates while Section 2.5 describes the methodology that was used for integration of the independent high fidelity ship motion model into the HLA federation and as such demonstrates a general way for integration of legacy non-HLA models into HLA based simulations. Sections 2.6 and 2.7 document the SOM and the FOM respectively. Details related to the federation execution are discussed in Section 2.8 and finally Section 2.9 provides a discussion of the performance of the RAS federation.

2.2 Introduction to HLA and RTI

The High-Level Architecture is a standard framework that supports simulations composed of different simulation components. The HLA was originally developed by the

Defense Modeling and Simulation Office (DMSO) of the U.S. Department of Defense (DoD) to meet the needs of defence-related modelling and simulation projects. Currently two separate HLA specifications exist: The earlier DMSO 1.3 specification [5] and the more recent IEEE 1516 standard [6]. The HLA was developed in order to facilitate the reusability of simulation models in different simulation scenarios and applications. In addition, the HLA supports interoperability, which is the ability to combine component simulations (federates) on heterogeneous distributed computing platforms, often with the goal of real-time operation. A collection of distributed federates operating together for a common purpose is referred to as a federation. For the federation to function effectively, three components are required:

1. HLA Rules: The rules ensure proper interaction of federates in a federation and describe the responsibilities of federates and federations.

2. Interface Specification: The interface specification defines Run-Time Infrastructure (RTI) services and interfaces, and identifies the "callback" functions that each federate must provide.

3. Object Model Template (OMT): The OMT prescribes the format and syntax for recording federation specific information and establishes the format of the following key models:
   (a) Federation Object Model (FOM);
   (b) Simulation Object Model (SOM); and
   (c) Management Object Model (MOM).

Two data types are defined by the HLA specification:

1. Object attribute updates are used to transmit the state of objects owned by a federate (for example, a vehicle's attributes might include position, orientation and speed). Attribute updates are sent only when attribute values change.

2. Interactions are used to notify federates about events generated or noticed by other federates (for example, a torpedo fired by a submarine federate).

Each federate publishes the attributes and interactions it can send and that can be of interest to other federates. Each federate subscribes to attributes and interactions published by other federates and that are of interest for that federate.

The Run-Time Infrastructure (RTI) is a collection of software libraries that implements the HLA interface specification. RTIs are available from several commercial vendors; usually in the form of software libraries with which the federate developer can link his/her federate code. Though all RTIs implement the HLA specification (either DMSO 1.3 or IEEE 1516) partially or in full, they differ significantly in their internal architecture and other implementation details. The services provided by the RTI software to the federates are:

1. Federation management: Creation, dynamic control, modification and deletion of federation execution.

2. Declaration management: Federates declare to the RTI their desire to publish and/or subscribe to object-state attributes and interactions.

3. Object management: Each object is assigned an ID that is used by the federate in order to transmit the object's state attributes.

4. Ownership management: Allows federates to transfer ownership of object attributes or complete objects since only the owner of a specific object instance may publish values for that instance.

5. Time management: Provides time-ordered exchange of events among federates in a federation.

6. Data Distribution Management (DDM): Supports efficient routing and distribution of data by abstract regions.

In this research only the following services were used: Federation management, declaration management, object management and time management. An HLA federation consists of between two and N distributed federates. Each federate is implemented as a user module that consists of software (code) and sometimes dedicated hardware as well. The federate interacts with other federates using RTI services exclusively. As illustrated in Figure 2, each federate's code (user code) generates calls to the RTI ambassador. The RTI ambassador interacts through the Local RTI Component (LRC) which is a RTI-specific interface (this unfortunately implies that an RTI from vendor A cannot interoperate "on the wire" with an RTI from vendor B, and might not even interoperate with a different version of the same RTI). The LRC on the other federate/federates then invokes callbacks via the local federate ambassador. The federate ambassador is essentially just a set of user (federate developer) implemented functions that are predefined by the HLA specification. The RTI can be configured and customized in many ways through a configuration file called the RID file (i.e.: network addresses are set using this file).

2.3 Time Stepped Federation

In order to meet the analysis requirements of DRDC Atlantic, it was determined that the RAS federation would use time-stepping as a method for controlling and sequencing events as they occur during the simulation execution. A time-stepped federation is inherently a Time Managed (TM) federation where all federates maintain the same sense of logical time during each simulation step. As such, the RAS federation uses the standard TM services provided by the RTI. In the RAS federation the execution manager federate acts as the scheduler for the federation.

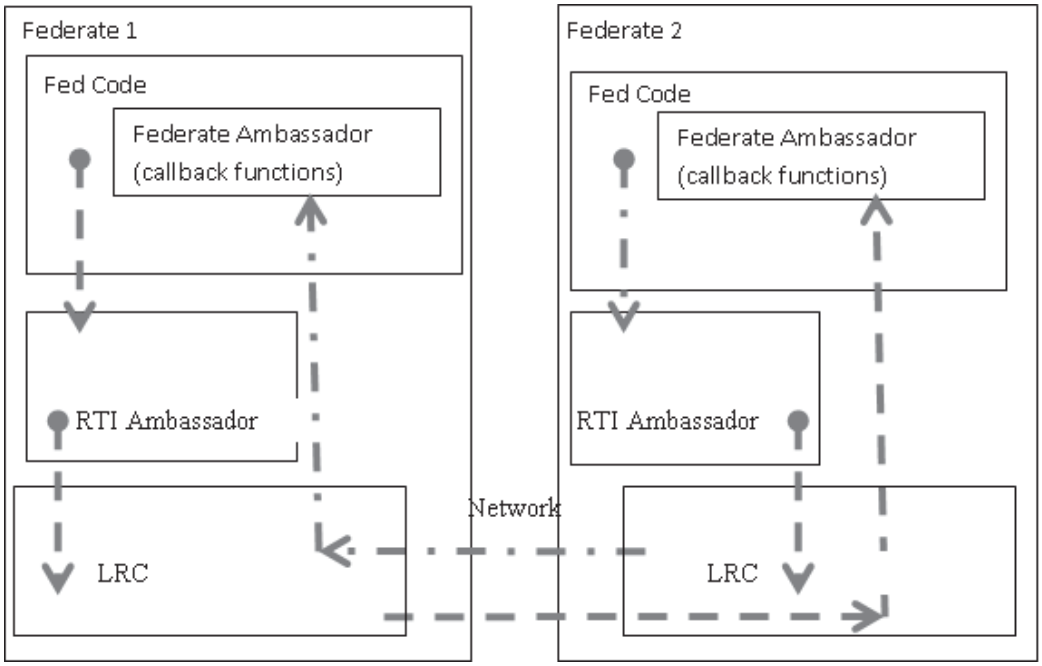

Figure 2: *HLA federate implementation.*

2.3.1 HLA Time Management

The baseline HLA provides two types of message ordering:

Receive Order (unordered): messages passed between federates with no guarantee that they will arrive in the same order in which they are sent.

Time Stamp Order (TSO): sender assigns a time stamp to the message; and successive messages passed to each federate have non-decreasing time stamps. Messages can be sorted by time stamp to ensure that messages are processed in the same order in which they were transmitted.

Table 1 shows a functional comparison between the two types:

In HLA, logical time is synonymous with simulation time and time advances by each federate must be properly managed to ensure no federate receives a message in its past. HLA Time Management (TM) services define a protocol for federates to advance their logical time and the RTI will ensure TSO messages are not delivered in a federate's past. This is a vital component of the RAS federation given the importance of accurate data exchange between the mathematical models acting as the simulation engines beneath each federate.

Table 1: *HLA message ordering.*

PROPERTY	RECEIVE ORDER (RO)	TIME STAMP ORDER (TSO)
Latency	low	higher
Reproduce before and after relationships?	no	yes
All federates see same ordering of events?	no	yes
Execution repeatable?	no	yes
Typical applications	Training, T & E	analysis

2.3.2 Time Regulating and Time Constrained Federates

Federates are defined within the time managed federation as time regulating and/or time constrained. Federates must declare their intent to utilize time management services by setting their *time regulating* and/or *time constrained* flags.

Time regulating federates:

- Can send TSO messages;
- Can prevent other federates from advancing their logical time;
- Enable Time Regulation; and
- Disable Time Regulation.

Time constrained federates:

- Can receive TSO messages;
- Time advances are constrained by other federates;
- Enable Time Constrained; and
- Disable Time Constrained.

Each federate in a federation execution can be:

- Time regulating only (e.g., message source);
- Time constrained only (e.g., Stealth);
- Both time constrained and regulating (common case for analytic simulations); or

- Neither time constrained nor regulating (e.g., DIS-style training simulations - also known as real-time).

Though in the RAS federation not all federates both send and receive TSO messages, for the sake of uniformity, all federates are configured to be time regulating and time constrained.

2.3.3 Related Object Management Services

The following object management services are used:

- Update Attribute Values $\Longrightarrow$ invokes a Reflect Attribute Values callback.
- Send Interaction $\Longrightarrow$ invokes a Receive Interaction callback.

The available message ordering types for each one of the message types above are Receive Order (RO) and Time stamp Order (TSO).

The ".fed file" defines the preferred order type for each type of object attribute and interaction. By default, all calls to *Update Attribute Values* or *Send Interaction* generate an RO message. The message will be sent as a TSO only if the sending federate is time regulating and a time stamp is attached to the message. A TSO message will be received as such only if the receiving federate is time constrained.

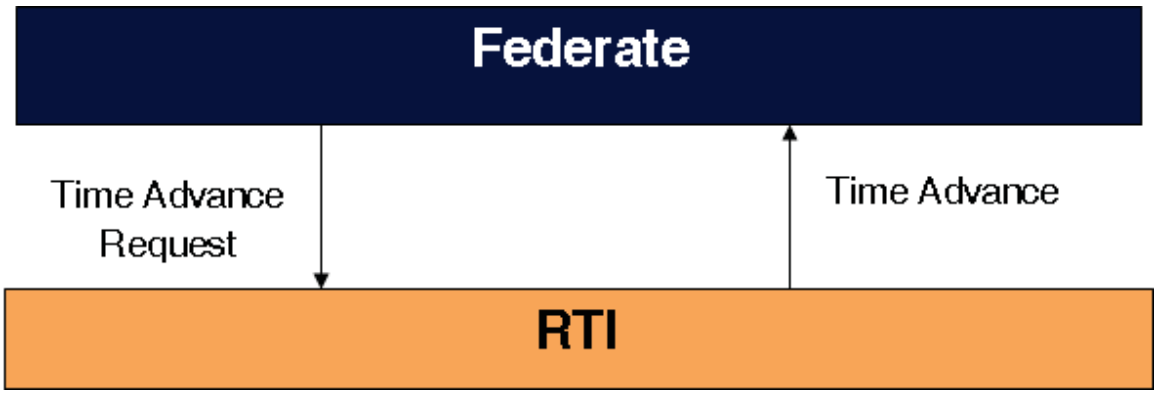

Figure 3: *Simulation time advance request/grant process.*

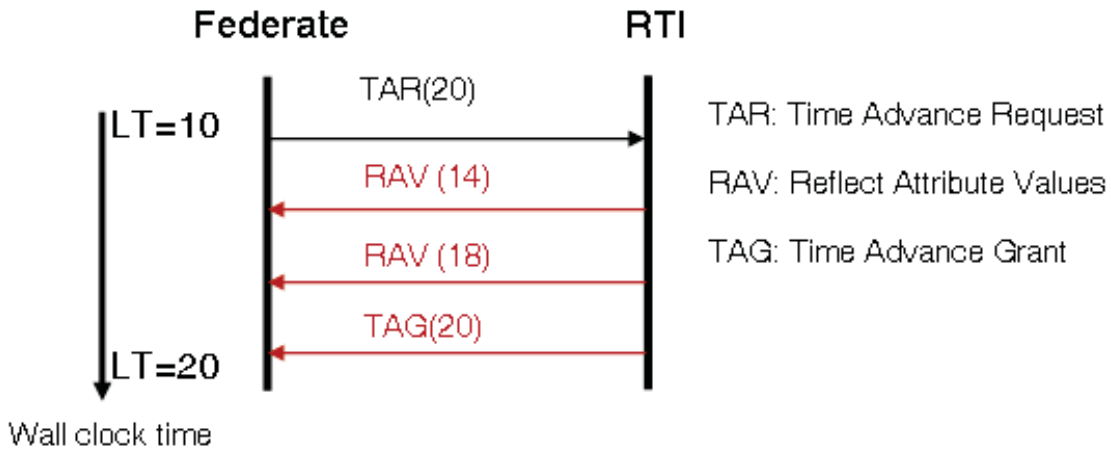

Figure 4: *Time stepped simulation, typical execution sequence.*

2.3.4 Time Advancement in Time Managed Simulation

HLA TM services define a protocol for federates to advance logical time. Logical time only advances when that federate explicitly requests an advance and the RTI determines that the entire simulation is ready for time advancement. Figure 3 illustrates the process of time advance request issued by the federate and the time advance grant provided by the RTI.

If the logical time of a federate is T, then the RTI guarantees no more TSO messages will be passed to the federate with time stamp$< T$. A more detailed description of Time Advance Requests (TAR) and Time Advance Grants (TAG) typically used by time stepped federates is provided in Figure 4.

In this process, the federate invokes Time Advance Request (TAR 20) to request its logical time be advanced to T (LT 20). The RTI then delivers all TSO messages with time stamp$\leq T$ (RAV 14 and RAV 18). Once complete, the RTI advances federates'

time to T, and invokes Time Advance Grant (TAG 20) when it can guarantee all TSO messages with time stamp$\leq T$ have been delivered.

Note that Grant Time always matches the requested time.

2.4 The Federates

The RAS federation consists of 10 federates with various levels of fidelity and complexity. The supply ship federate and the receiving ship federate are based on the same ship motion engine but with different configuration parameters. Both helm federates (supply ship helm and receiving ship helm) are identical in implementation.

2.4.1 Execution Manager Federate

This federate monitors and controls the general execution of the simulation system and has 6 major roles:

1. Ensure that all participating federates have joined the federation before the simulation can start.
2. Bring all federates to a known state before the simulation starts (through synchronization points).
3. Ensure that all federates are using the same seaway representation (same seaway definition file based on MD5 signatures).
4. Pace the execution based on wall clock time (real-time, faster than real-time or slower than real-time).
5. Stop execution and insure that all federates resign from the federation before the federation is destroyed.
6. When a new configuration (start up conditions) is required, halt the simulation, then manage the simulation in the new configuration.

2.4.2 Ship Motion Federates (Supply and Receiving Ships)

The ship motion federates are based on a high fidelity ship motion model called ShipMo3D that was developed at DRDC Atlantic. The model was integrated into the RAS federation using a "wrapper" federate as described in Section 2.6. The ship motion federate receives inputs at each time step that include the RAS gear force and moment vectors as computed by the RAS gear model and produces outputs in the form of kinematic data that is used by the RAS gear model during the next time step. The kinematic data is represented using ShipMo3D's native coordinate system

described below. This output data is also used by the visualization federate to update a visual representation of the two ships. In addition, this data is logged by a data logger federate. The ship motion federate is capable of accepting hydrodynamic forces from an external source (such as a hydrodynamic force federate) as well as from an external seaway configuration file. At present, external hydrodynamic forces are not applied and seaway data is defined via a configuration file.

Ship and payload motion data are passed through the federation using coordinate systems consistent with the ShipMo3D ship motion simulation package [7] used for generating time-varying ship motion of both the supply and receiving ships. ShipMo3D uses a translating earth coordinate system to represent ship motions and an earth-fixed coordinate system to track ship position. The motions generated by ShipMo3D, and ultimately consumed by the RAS federation, are with respect to the vessel's calm water orientation. It is important to note that the calm water position and attitude, from the perspective of the RAS federation, must incorporate the ship's draft, trim, and centre of gravity position. Outputs from the federation are published relative to the earth-fixed coordinate system such that translational coordinates locate the ship centre of mass relative to the origin of the earth fixed system having an elevation corresponding to the centre of gravity position in calm water. Roll and pitch orientations are specified relative to the calm-water orientation and absolute heading is published where 0 degrees corresponds to north and 90 degrees to east. Absolute payload orientations are published relative to the earth-fixed coordinate system.

2.4.3 Ship Helm Federates (for Supply and Receiving Ships)

The ship helm federates provide a basic interface for providing human in the loop control of the ship motion models. The operator can update the ship's propeller RPM as well as autopilot heading. The helm federate will display to the operator the actual RPM, heading, speed and rudder angle as provided by the ship motion federate (calculated by the ShipMo3D model). Additional operator's inputs can be easily added to the helm federate if required. At present the helm federate reads the operator's inputs from a text file and displays data to the console. The helm inputs are sent by the helm federate at each time step and are available to the ShipMo3D model to be used at the next time step. The data is resent at each time step independent of any changes made by the operator. The current data from the ship motion federate is updated at the helm at each time step as well. The helm inputs are also received by the data logger federate. All helm data in the federation is expressed in earth-fixed and translating earth coordinate systems as described in Section 3 of Reference [7]. The full ship helm federate is developed in C++ using Microsoft Visual Studio .net and is statically linked against the RTI libraries.

2.4.4 RAS Gear Federate

The RAS gear federate consists of a high fidelity computational model (for detailed description please refer to Section 3) implemented in Fortran and a wrapper federate that generates calls to the computational model (from C++ to Fortran). The RAS gear federate receives ship spatial data from the ship motion federate. This data consists of 6 DOF position, velocity and acceleration parameters for both ships. The federate publishes the 3D forces expressed in the earth-fixed coordinate system and moments computed for each ship expressed in their respective local ship-fixed coordinate systems. In addition, the federate outputs cable tensions, 6 DOF spatial data for the payload (position, velocity, and acceleration), and 3D position for a requested number of points along the length of the RAS highline cable (in earth-fixed coordinates). This data is used by the visualizer federate to display payload and cable segments. All the data is also logged by the data logger federate. The computational model is developed in Fortran using Intel's Fortran compiler version 9.1 to generate a software library. The wrapper federate is developed in C++ using Microsoft Visual Studio .net and is statically linked against the RTI libraries and the computational model (Fortran) library.

2.4.5 Data Logger Federate

The data logger federate subscribes to the entire volume of data that is exchanged between the RAS federation federates. This federate does not send any data over HLA, but at each time step logs the most current data into a set of text files that can be later utilized for post execution analysis. Each data line in each of the log files corresponds to a single time step in the federation's execution and the first column is always the time stamp. A detailed description of all output files generated by the data logger is provided in Annex A. This federate is developed in C++ using Microsoft Visual Studio .net and is statically linked against the RTI libraries.

2.4.6 Seaway Federate

The seaway federate is a placeholder which is envisioned to provide seaway data to the ship motion federate at some future point in federation development. At present this is a null federate: it joins the federation and then calculates and sends the MD5 signature of the seaway definition file to the execution manager federate. The MD5 signature is used by the Execution Manager federate and ship motion federates to ensure that all federates are using a consistent seaway definition. From there, it advances in time with the rest of the RAS federation but does not publish any data. The seaway data is currently provided to the ShipMo3D model using a configuration file that is generated by the SM3DbuildSeaway application as described in Reference [8] Section 7. This federate is developed in C++ using Microsoft Visual Studio .net

and is statically linked against the RTI libraries.

2.4.7 Visualizer Federate

The visualizer federate subscribes to the ships spatial data as well as the pay-load spatial data and cable point positions. The federate accepts the spatial data at a rate of 10 Hz and will use dead-reckoning to provide smooth visualization at a rate of 50 Hz (minimum).

2.4.8 Hydrodynamic Interactions Federate

The hydrodynamic interaction federate is a placeholder federate that is envisioned to provide the ship motion federates with hydrodynamic forces at some future point in federation development. It is envisioned that this federate will calculate hydrodynamic forces based on ship dynamics data (i.e. ships spatial data). At present this is a null federate: it joins the federation and advances in time with the rest of the RAS federation but does not publish any data. This federate is developed in C++ using Microsoft Visual Studio .net and is statically linked against the RTI libraries.

2.5 Time Management in the RAS Federation

As discussed in Section 2.3, HLA federates can be time regulating, time constrained or both depending on whether or not they have a requirement to send or receive TSO messages. Table 2 summarizes the TSO messages for each federate.

As shown in Table 2, most federates both send and receive TSO messages, hence they are required to be both time regulating and time constrained. The Data logger federate can only be time constrained while the seaway federate can only be time regulating. The federation manager federate serves as a scheduler for the federation and hence must be time regulating and time constrained. At present the visualizer federate runs as a playback federate but is intended ultimately to execute as a time constrained federate. For the sake of uniformity and reuse of code both the seaway federate and the data logger were set as time regulating and time constrained since the performance impact of such a setup was found to be minimal.

2.5.1 Time Management Parameters

In the RAS federation all federates use a time step of 0.1 logical time units (equivalent to 0.1 seconds for this federation). This time step was selected for optimal function of the ShipMo3D ship motion model and the RAS gear model that each require a time step of 0.1 seconds. Since all federates use a time step of 0.1 logical time units, the LookAhead parameter is set to 0.1 logical time units as well for optimal performance

Table 2: *RAS federation TSO messages per federate.*

Federate	Sends TSO message	Receives TSO message
Federation manager	N/A	N/A
RAS gear federate	RAS forces, cable segments, payload spatial	Ships' spatial
Supply ship	Ship's spatial, propeller RPM, rudder angle, linear speed, actual heading	RAS forces, helm inputs
Receive ship	Ship's spatial, propeller RPM, rudder angle, linear speed, actual heading	RAS forces, helm inputs
Data logger	N/A	All data sent by RAS gear, supply ship, receive ship and helms
Supply ship helm	Autopilot requested heading, requested RPM	Actual RPM, rudder angle, linear speed, actual heading. All from supply ship federate.
Receive ship helm	Autopilot requested heading, requested RPM	Actual RPM, rudder angle, linear speed, actual heading. All from receive ship federate.
Hydrodynamic interactions federate	Hydrodynamic forces	Ship's spatial.
Vizualizer federate	N/A	Ship's spatial, payload spatial, cable positions
Seaway federate	Seaway parameters	N/A

(a smaller LookAhead value would have increased overhead without providing any performance benefit).

2.5.2 Pacing the Federation Execution

The Execution manager federate can associate a logical time step of 0.1 units with 0.1 seconds of wall clock time in order to achieve real-time simulation. It can also associate the same logical time step with a shorter interval of wall clock time for faster than real time simulation or it can associate it with a wall clock interval of more than 0.1 seconds to achieve slower than real-time simulation.

2.6 Integration of ShipMo3D into the RAS Federation

ShipMo3D was developed by DRDC Atlantic as a high fidelity ship motion model. It receives as inputs: seaway data, hydrodynamic forces and external forces applied to the ship as well as helm controls (RPM, rudder, etc.) and provides spatial data (position, velocity, acceleration) in earth-fixed coordinates (please refer to Section 3 of Reference [7] for a detailed description) as output. The spatial data is distributed across the federation without any coordinate conversion. The ShipMo3D model consists of a large number of modules, all of which are implemented in the Python language to utilize the extensive mathematical libraries available to Python users. Since there are no RTIs available that interface natively with Python code, a Python federate was created. The federate is written in Python to interface with ShipMo3D and implements a minimal set of federate ambassador calls required by the RTI. Using the SWIG (`http://www.swig.org`) tool-chain, a wrapper DLL is generated. This DLL enables calling a subset of the LRC functions from Python when going down stream (federate to RTI). In the upstream direction (RTI to federate) the RTI library invokes the federate ambassador through a Python wrapper.

2.6.1 Calling Federate Ambassador Callbacks from the RTI

The RTI initially calls a standard federate ambassador function implemented in C++. The C++ function then calls a "stub function" that is implemented solely for the purpose of linking with C++ code. In reality, the function that is executed is part of `\_WrapRTI\_MAK\_1516.DLL`, which is generated by the SWIG tool chain using a SWIG interface file that must be provided by the user. The library `_1516.DLL` will invoke the Python implementation of the callback that is part of the Python federate ambassador.

2.6.2 Calling the RTI from the Python Federate

The Python federate generates a call to an interface function that is part of `_WrapRTI_MAK_1516.DLL`. That interface function then calls a C++ interface function with the same name. The C++ interface function that is implemented in the `SwigRTIAmbassador\_MAK\_1516.cxx` file then calls the actual RTI function.

2.7 Federation Object Model (FOM) and Base Object Model (BOM)

The RPR FOM 2.0 draft 17 is a recognized industry standard federation object model for use in platform level simulations. It is compliant with the IEEE HLA 1516 specification and forms the basis for the development of the RAS FOM. This

helps to ensure easy integration with other HLA compliant simulation systems and federations. The RAS FOM was created by extending the RPR FOM through the addition of those classes and attributes required by the RAS federates using the mathematical models developed as part of this project.

Tables 3 and 4 provide a summary of all the FOM classes (and attributes) used in the RAS federation.

Table 3: *Outline of RAS FOM object class structure.*

Object Class	Attribute	Function/Comment
RasGearVectors	Feqssp	force vector equipollent acting on supply ship given in inertial frame
	Meqssh	moment vector equipollent acting on supply ship in inertial frame
	Feqrsp	force vector equipollent acting on receiving ship in inertial frame
	Meqrsh	moment vector equipollent acting on receiving ship in inertial frame
	Thl	actual highline cable tension
	Tih	actual in-haul cable tension
	Toh	actual out-haul cable tension
SeaWay	Amplitude	waves amplitude
	Frequency	waves frequency
	Phase	waves phase
	Heading	waves heading
HelmInputs	helmAutoPilotHeading	requested heading
	helmRpm	requested RPM (all propellers)
	CmdRudderAngle	requested rudder angle
	helmMode	requested helm mode (rudder control or autopilot control)
HydroInteractions	FSupply	force vector imposed on supply ship
	MSupply	moment vector on supply ship
	FReceive	force vector on receiving ship
	MReceive	moment vector on receiving ship
PayLoad	Spatial	Payload position, including angular acceleration.
	PayLoadIdentifier	string to identify the payload
CablePoint	Spatial	position of cable point (3D vector)
	Index	unique index of point on cable
BaseEntity .PhysicalEntity .Platform .SurfaceVessel	Spatial	Spatial data for ships, including angular acceleration
	ShipRpm	RPM of propeller 1
	ShipHeading	ship's heading
	ShipRudderAngle	rudder angle in deg
	ShipSpeed	magnitude of ship's velocity vector

Table 4: *Outline of RAS FOM interaction class structure.*

Interaction Class	Parameters	Function/Comment
SeaWayFile	FedName	federate's name
	MD5Signature	seaway file MD5 calculated signature
RasManagement	JoinStatus	status enumeration
	ExecutionCommand	command enumeration
	FedName	federate enumeration

The following describes the classes, attributes, interactions, and parameters that were added (or modified) in the RPR FOM to create the RAS FOM.

RasGearVectors is a new class added to facilitate the forces applied to the ships by the RAS-gear (see Figure 5). Feqssp (*F*orce *eq*uipollent *s*upply ship *sp*ace frame), Meqssh (*M*oment *eq*uipollent *s*upply ship *sh*ip frame), Feqrsp (*F*orce *eq*uipollent *r*eceiving ship *sp*ace frame) and Meqrsh (*M*oment *eq*uipollent *r*eceiving ship *sh*ip frame) are 3D vectors of double precision (8 bytes). Thl (Tension *h*igh*l*ine), Tih (Tension *i*n-*h*aul) and Toh (Tension *o*ut-*h*aul) represent cable tensions in Newtons and are of double precision each. Feqssp and Feqrsp are represented in the same inertial coordinate system that is established by the generated ship motion. Meqssh and Meqrsh are represented in the coordinate system of the supply and receiving ships respectively (ship-fixed) where the coordinate system is right handed with x forward and z up. Consistent with accepted best practice working with Newton-Euler equations, forces are expressed in the inertial coordinate system to allow the gravity vector to remain constant in applying Newton's equation; and moments are expressed in body-fixed coordinates to allow the use of Euler's equation (and to avoid a requirement to continually transform the rigid body inertia matrix when using the more general dynamic moment balance formulation). All forces are given in Newtons and all moments are given in Newton-metres.

RasGearVectors
-Feqssp -Meqssh -Feqrsp -Meqrsh -Thl -Tih -Toh

Figure 5: *Attributes of the RasGearVectors Class.*

The **SeaWay** class is a new class, added to facilitate seaway inputs to both ship federates (see Figure 6). All 4 attributes are vectors of floats (4 bytes). Amplitude is presented in metres, frequency in Hz, phase in radians and heading in degrees where north is 0.0.

HelmInputs is a new class added to facilitate the helm inputs provided to the ship motion model by the operator (see Figure 7). At present two inputs are supported: auto pilot heading that is passed in degrees as a float (4 bytes) where 0.0 degrees is north and propeller RPM that is a float as well. This RPM will be applied to any number of existing shafts on the vessel. CmdRudderAngle is a float (4 bytes)

SeaWay
-Amplitude -Frequency -Phase -Heading

Figure 6: *Attributes of the SeaWay Class.*

that represents rudder angle in relation to ship's hull in degrees. The helmMode attribute indicates whether the rudder is controlled based on input commanded rudder angle (CmdRudderAngle) or autopilot. "helmMode" is an enumeration where: 1-use autopilot, 2-use rudder angle.

HelmInputs
-helmAutoPilotHeading -helmRpm -CmdRudderAngle -helmMode

Figure 7: *Attributes of the HelmInputs Class.*

The **HydroInteractions** class is a new class, added to facilitate hydrodynamic forces (Fsupply, Freceive) and moments (Msupply, Mreceive) imposed on each ship by the other ship (see Figure 8). All forces and moments are represented by 3D vectors where each component is represented by a double (8 bytes). Fsupply and Freceive are represented in the same inertial coordinate system that is established by the generated ship motion. Msupply and Mreceive are represented in the coordinate system of the supply and receiving ships respectively where the coordinate system is right handed with x forward and z up. All forces are given in Newtons and all moments are given in Newton-metres.

HydroInteractions
-FSupply -MSupply -FReceive -MReceive

Figure 8: *Attributes of the HydroInteraction Class.*

The **Spatial** attribute of **BaseEntity** was extended by adding the angular acceleration vector (see Figure 9). Please refer to References [4, 6] for a detailed description of standard RPR-FOM attributes.

BaseEntity
-EntityIdentifier.*
-EntityType.*
-IsPartOf.*
-RelativeSpatial.*
-Spatial.*
-Spatial.AngAccelVec

Figure 9: *Attributes of the BaseEntity Class.*

The **SurfaceVessel** class inherits from class **Platform**, but was extended by adding (1) the ship's actual propeller RPM (for shaft number 1), (2) the ship's actual heading rudder angle in degrees with relation to the ship's hull and (3) the speed in metres/second as calculated by the ship model (see Figure 10). Note that the ship's propeller RPM and heading as reported by the ship's model may differ from the values requested by the helm's operator. All four attributes are represented as floats (4 bytes).

SurfaceVessel
-ShipRpm
-ShipHeading
-ShipRudderAngle
-ShipSpeed

Figure 10: *Attributes of the SurfaceVessel Class.*

The **PayLoad** class is a new class that was introduced to represent the payload (see Figure 11). The payload object will be present in the federation for both data logging purposes and to provide an input to visualization. It contains two attributes, the identifier that is a string of characters and can be used by the visualizer in order to associate the proper visual representation of the payload when passed between the ships. The **Spatial** attribute is identical to the extended **Spatial** attribute of the **BaseEntity**. A detailed description of the **Spatial** attribute is available in References [4, 6].

PayLoad
-PayLoadIdentifier -Spatial

Figure 11: *Attributes of the Payload Class.*

The **CablePoint** class represent a single point on the highline cable (see Figure 12). Each cable point object has an index (four byte integer) and a Spatial attribute. At present only the position portion of the **Spatial** attribute is populated. The cable points divide the highline cable into $N-1$ segments (where N is given as a configuration parameter for the RAS gear federate and represents the total number of cable points). Highline cable segment n begins at cable point indexed n and ends at cable point indexed $n+1$. Cable point 0 is the attachment point on the supply ship while cable point $N-1$ is the attachment point on the receiving ship (cable points are indexed 0 to $N-1$ and cable segments are indexed 0 to $N-2$).

CablePoint
-Index -Spatial

Figure 12: *Attributes of the CablePoint Class.*

RasManagment is a new interaction class which was added to facilitate simulation management data exchange between the execution manager federate and the other federates (see Figure 13). JoinStatus is an enumeration (1 byte) where: 1-Announce federate, 2-federate announcement, 3-All federates joined, 4-All quit. Execution command is an enumeration (1 byte) where: 1-Start execution, 2-Reload configuration. FedName is an enumeration (1 byte) with a unique value for each federate where: 10-Manager, 11-RAS gear, 12-Supply ship, 13-Receiving ship, 14 - Helm for supply ship, 15 - Helm for receiving ship, 16 - Data logger, 17-Seaway, 18-Hydrodynamics interactions.

RASManagement
-JoinStatus -ExecutionCommand -FedName

Figure 13: *Attributes of the RASManagement Class.*

SeaWayFile is a new interaction class which was added to ensure that all federates are using the same seaway definition file (see Figure 14). Since federates may reside on different machines with different directory structures, the seaway definition file is identified by its MD5 signature that is locally calculated by each federate and then sent to the execution manager for verification. FedName is an enumeration (1 byte) with a unique value for each federate where: 10-Manager, 11-RAS gear, 12-Supply ship, 13-Receiving ship, 14 - Helm for supply ship, 15 - Helm for receiving ship, 16 - Data logger, 17-Seaway, 18-Hydrodynamics interactions. MD5 Signature is an ASCII string of variable length that holds the MD5 signature of the seaway definition file.

SeaWayFile
-FedName -MD5Signature

Figure 14: *Attributes of the SeaWayFile Class.*

2.8 Simulation Execution

The execution of the RAS simulation system can be broken out into four major phases:

1. Create the federation and allow all federates to join.
2. Perform all pre-time stepped data exchange to bring all variables to a ready state.
3. Execute the simulation steps beginning from predefined initial conditions for a desired duration.
4. Gracefully resign from the federation and destroy the federation.

Note that a detailed description of all configuration files and parameters for all federates is provided in Section A.

2.8.1 Phase 1 – Create Federation

Each federate that attempts to join the federation will first attempt to create the federation by calling `createFedEx()`. This will ensure that a federation is created regardless of which federate starts first. Only the first federate to join should be successful in creating the federation and all subsequent federates will receive the message that the FedEx already exists and the call will generate a soft exception. This soft exception will have no impact on the federation execution or the federate. After attempting to create the federation, each of the federates will join the federation by making a call `joinFedEx()`.

Part of the process of joining the federation is the initialization of data and establishing which data the federate is going to publish and subscribe to. This data can be either an "object" or an "interaction". In the RAS federation "interactions" are used for execution management and hence require all federates to subscribe to those "interactions".

During the creation and initialization of the federation it is the Execution Manager Federate's responsibility to ensure that the federation achieves a state where it is ready to start the simulation. Each federate waits for instructions from the Execution Manager and responds to these requests until the Execution Manager federate signals that all federates are joined and that the simulation is ready to proceed. As soon as all federates are synchronized, a call to `synchronizationPointAchieved()` is made and the federation is ready to advance to Phase 2.

2.8.2 Phase 2 - Pre-time-managed Steps

Since all data exchange during the time managed phase of execution in the simulation should follow a very strict sequence during each simulation step, a pre-time-managed phase was added in order to allow for certain steps to take place and properly initialize all values and state variables in preparation for the time stepped execution phase. Specifically during this phase, the RAS gear federate first receives the spatial data for both ships. Based on this data the RAS gear can initialize its logic to a known state and produce an initial set of forces that are sent to the ship motion models so the ship motion models can initialize their variables to correct start conditions. In addition, during this phase the MD5 signatures for seaway definition files are being exchanged and verified for consistency. At the end of this phase all federates reach a second synchronization point, time management is turned on and the federation will advance into the third time managed phase of simulation.

2.8.3 Phase 3 - Time Managed Simulation Execution

In the time managed phase the calculated results or data from time step t_i is being used as the input (or part of the input) for the calculations during time step t_{i+1}, hence the federates that use results from time step t_i to compute during time step t_{i+1}, would not advance to time step t_{i+1} before data with a time stamp of t_i is available for them from all data sources (federates) they are dependent upon. Federates that do not depend on data from other federates for their computations (or simple federates such as the data logger which does not perform any computations) still would not be able to advance from time step t_i to time step t_{i+1} since they are time constrained.

During each simulation time-step three concurrent activities take place:

1. The RAS-Gear federate uses spatial data from both ship-motion federates from the previous time step and calculates force vectors, which it then distributes to the ship-motion federates for use in the next time step.

2. Both ship-motion federates use the force vectors generated by the RAS-gear and the helm federates during the previous time-step as input for calculating the latest spatial data. Then each ship-motion federate distributes the new spatial data to be used by the RAS-Gear federate and the helm federate during its next time-step.

3. The helm federate uses the data from the ship motion model (rudder angle, RPM, heading, speed) to display latest values. Then if new RPM and autopilot heading are available from the operator, the helm federate will distribute them to the ship motion federate to be used during the next time step. If there are no new values available, the helm federate will distribute the most current values.

Once the five federates (RAS-Gear, supply ship-motion, receiving ship-motion, helm for supply ship , helm for receiving ship) send their data, they each make a request to move to the next time-step by calling `timeAdvanceRequest()`. By default, the data logger federate, the sea way federate and the execution manager federate generate the same request as well. This request is granted by the RTI once two conditions are met:

- all federates have published the results of their respective calculations, and
- all federates are requesting to advance to the next time step.

In this way, the execution-manager federate can pace the simulation by issuing a `timeAdvanceRequest()` at given wall-clock time intervals. The Execution Manager federate measures a given wall-clock time period and places a time advance request to the next simulation time-step. In addition, the Execution Manager acts upon such

operator input as requests to terminate or restart the simulation. When a request to terminate the simulation is issued to the Execution Manager, it initiates Phase 4.

Note that the visualizer federate at this point is not time managed and simply updates the visual display as new spatial data are received or based on extrapolation from previous data (dead reckoning).

2.8.4 Phase 4 - Resign and Destroy Federation

When a request to terminate the simulation is received, the Execution Manager federate sends an interaction to all federates to terminate the simulation. Federates quit the simulation loop and move to a synchronization point. When all federates have achieved this point they resign from the federation by calling `resignAndDestroy()`. The result of this call is the destruction of the federation by the last federate to resign.

2.9 Federation Performance

The RAS federation consists of 9 time managed federates and one non-time managed federate (visualizer). When all 9 time managed federates shared a single PC with a single 1.8 GHz processor and 512 MB of RAM the federation was capable of completing 9 time steps per second. As will be shown in the following section, the major factor is the cost of time management rather then the complexity of calculations performed by the two ship motion models and the RAS gear.

2.9.1 Calculation of Grant Time

From the description in the previous section, one can see that in order for the time managed federation to advance to the next step, the system must wait until all federates have completed their calculations before processing all time advance requests initiated by the federates and the corresponding grants given by the RTI. It is easy to see why the use of time management in a federation can be the cause of reduced performance of the simulation system as a whole. This reduction may be due to the simple overhead of making time advance requests and grants between each federate and the RTI but may also be due to the interval of logical time, which is driven by the calculation of the Lower Bound on Time Stamp (LBTS). The following considerations should be made with respect to the LBTS:

1. The LBTS value computed for a federate is the lowest (earliest) time stamp of messages that are destined for that federate later in execution;

2. For each federate, the RTI must ensure that:

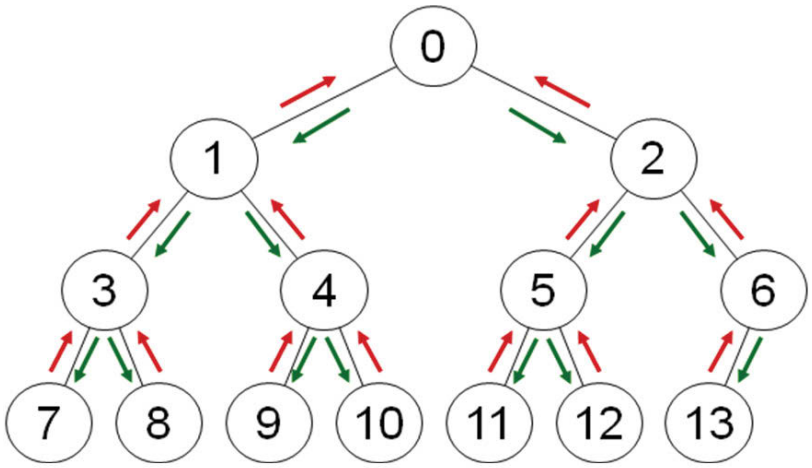

Figure 15: *Example of the tree barrier process.*

i Time Stamp Ordered (TSO) messages are delivered to the federate in time-stamped order, and

ii No message is delivered to the federate with a time stamp that is earlier than its logical time.

3. Once the LBTS for a federate is computed:

iii The RTI delivers to the federate all TSO messages with a time stamp earlier than LBTS, and

iv If the RTI prevents the federate from advancing its logical time beyond LBTS, it guarantees that the federate receives no messages in its past.

4. To compute the LBTS, the RTI must consider:

v The earliest time stamp of a TSO message that any federate might generate in the future (the current logical time of a federate is one boundary since no federate can generate a TSO message in its past), and

vi The time stamps of messages within the RTI and the interconnection network (transient messages).

All commercially available RTIs use the butterfly scheme for LBTS computation and, given their proprietary nature, are not modifiable by the developer or user of the system. The butterfly barrier scheme organizes processes into trees which are computationally superimposed over one another. At the individual node level, the process proceeds as follows:

1. The process sends a message to its parent process when the process has reached the barrier point;

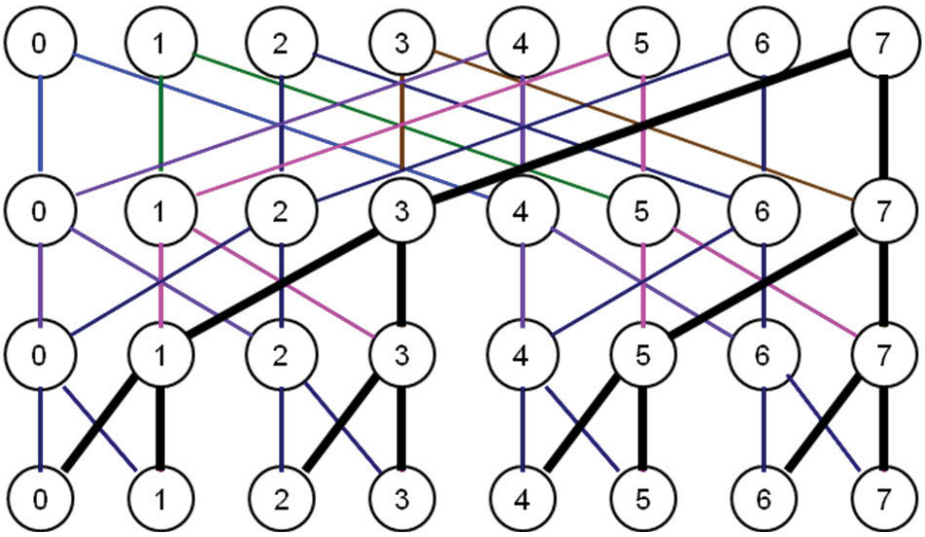

Figure 16: *Example of the butterfly barrier process.*

2. The root detects completion of the barrier when a message has been received from each of its children processes; and

3. The root broadcasts a message down the tree to release processes ($2 \log N$ time if all processes reach the barrier at same time) to advance to the next step.

The butterfly barrier process can be considered to be a superposition of a number of tree barrier processes (see Figures 15 and 16). A single LBTS computation is initiated when a federate requests a time advance and requires $N \log N$ messages to be exchanged between the federates, where N is the sum of time-regulating and time-constrained federates in the federation. Actual LBTS calculations may require between $N \log N$ and $(T + N) \log N$ messages (where T is the number of transient messages in the system when the LBTS calculation is initiated), where each message contributes its own message latency to the overall LBTS calculation. Some of these messages can be processed concurrently but a significant degree of serialization will be present and will be proportional to the depth of a binary tree with a federate at each node. For example, in a federation of 8 federates that are all time-regulating and time-constrained, and which are interconnected over a LAN with average network latency of 0.01 seconds, the minimal delay for a federate to advance to the next point in time is: $\log 8 \times 0.01$ sec $= 30$ milliseconds (the amount of serialization is $\log N$ not $N \log N$).

2.9.2 Performance of the RAS Federation

As mentioned above the federation is capable of 9 time steps per second. Since each time step represents 100 ms of wall clock time, the RAS simulation performs at

0.9 real-time. Redistribution of the RAS federation over several PCs may improve performance due to lower processor utilization as long as the LAN has a low enough latency. The visualizer federate does not affect the federation's performance since it is passive and non-time managed, and does not share a processor with the rest of the federates.

3 Mathematical Modelling of RAS Gear

3.1 Overview

Physical modelling of RAS equipment considers three bodies, three cables for which tension is actively controlled, and the behaviour of two sets of cable tension control equipment. Various entities comprising the multibody dynamic system are illustrated schematically in Figures 17, 18, and 19. The bodies consist of:

1. the supply ship;
2. the receiving ship; and
3. the payload.

The cables consist of:

1. the highline cable from which the payload is suspended;
2. the inhaul cable that runs from the supply ship to the payload; and
3. the outhaul cable that runs from the supply ship to the receiving ship, around a sheave, and back to the payload.

Cable tension control equipment is located on the supply ship and comprises:

1. the passive highline ram tensioner, illustrated schematically in Figure 19, and the associated actively controlled highline winch; and
2. the inhaul/outhaul cable tension control system that comprises two actively-controlled winches.

For the purpose of dynamic modelling, cable tensions are evaluated from the highline tension control and traversing system models and are considered to act directly between attachment points; in particular, between two points on the payload body and three attachment points on each ship. The tension control equipment is modelled in detail as it affects the cable tension outboard of the supply ship but specific forces or cable runs inboard of the ship attachment points are not explicitly considered apart from determining the total highline cable length as that total length is relevant for the calculation of cable stretch. The cable tensions result in an equipollent force and moment pair acting on each ship. The cable tensions also result in forces and moments acting on the payload body that enter into the payload governing dynamic equations of motion.

Overall, the model includes the six degree-of-freedom payload dynamics resulting in 12 first order differential equations; a model of the highline cable and ram tensioner

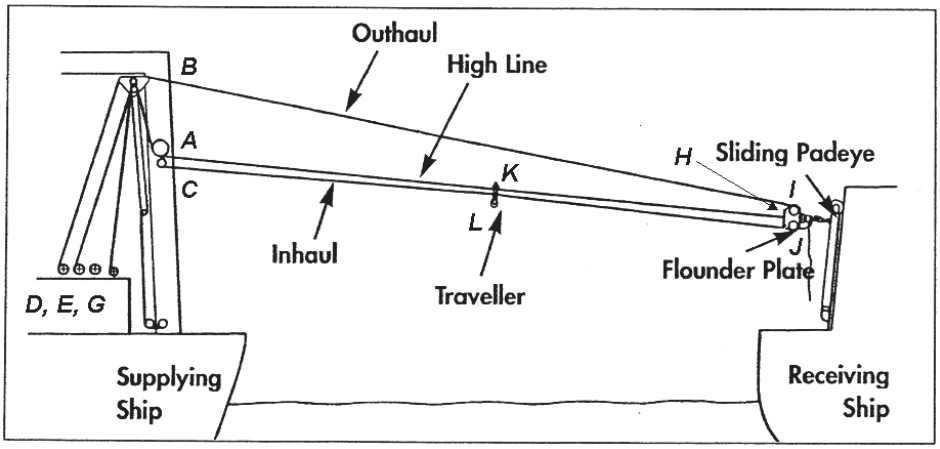

Figure 17: *Schematic illustration of ships and cables comprising the RAS system (Adapted from [8]).*

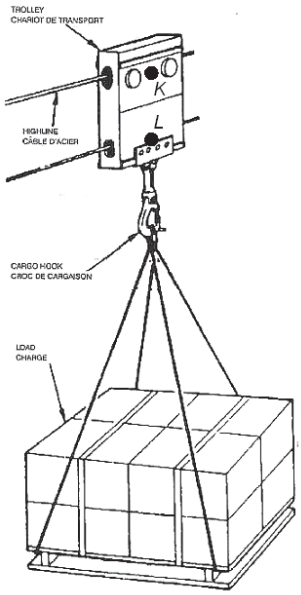

Figure 18: *Schematic illustration of payload body and its interface to the highline and traversing cables (Adapted from [9]).*

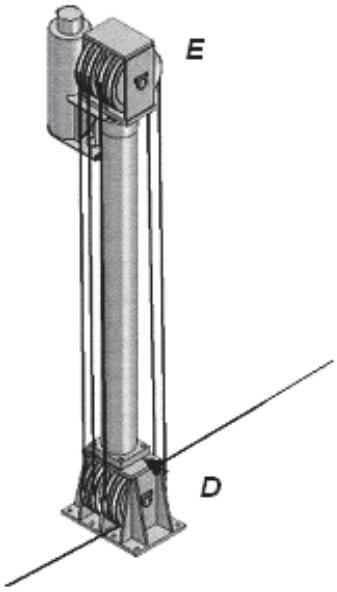

Figure 19: *Schematic illustration of a typical passive ram tensioner unit (Adapted from [10]).*

where the single degree-of-freedom is the total active cable length outboard of the highline winch and is represented by a single first-order differential equation; and a model of the in-haul and out-haul cable tensions that does not strictly have any degrees of freedom but rather directly generates cable tensions based on the difference between actual and commanded payload transfer velocity and in-/out-haul system commanded pretension. As described, the overall physical RAS system model would include 13 first-order differential equations. However, in addition, the cumulative ram travel from its neutral position is formulated as a differential equation and integrated consistently with the others such that it is available for use in a drift control algorithm that automatically adjusts outboard highline cable length in response to low frequency changes in the relative ship positions. Therefore, a total of 14 first-order differential equations are numerically integrated by the RAS gear model. The overall system mathematical model is implemented in a robust form such that it can easily be refined, if necessary, once its initial implementation is complete and its performance is evaluated, ideally against experimental data. The modelling philosophy has been to mathematically capture the system behaviour without encumbering the model with unneeded complexity.

3.2 Theoretical Development

3.2.1 Geometry

Points

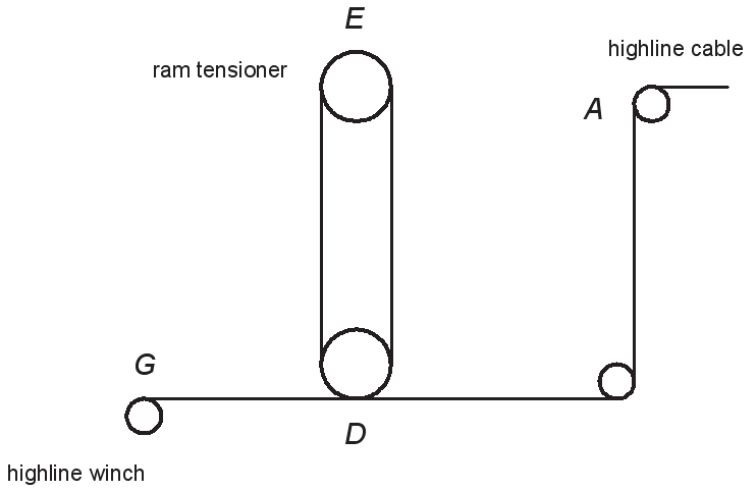

Figure 20: *Schematic illustration of relevant in-board cable runs with end points identified by points.*

The geometry of the RAS gear problem has been defined using a series of 11 key points labelled A through L[1] and identified in Figures 17 through 20. Points A, B, and C represent the interface points between the supply ship and the highline, out-haul, and in-haul cables respectively. Points H, I, and J represent the interface points between the receiving ship and the highline cable, out-haul cable attachment to the supply ship, and out-haul cable attachment to the payload respectively. Points K and L represent the interfaces between the payload and the highline and in-/out-haul cable systems respectively. Points D, E, and G are used for identifying relevant in-board points on the supply ship as illustrated in Figure 20. Additionally, each of the body centres of mass is labelled using points S, R, and P for the supply ship, receiving ship, and payload bodies respectively. Using these points, all necessary absolute and relative position and velocity vectors as well as essential unit vectors can be defined. The points are also used for representing individual lengths of cable that are required by the dynamic models as well as their first time derivatives. As examples, $\vec{r}_{AK}$ is the relative position vector from the highline attachment point on the supply ship (Point A) to the highline interface with the payload (Point K); $\vec{v}_{AK}$ is the corresponding relative velocity vector; and L_{AK} is the length of highline cable between the supply ship and the payload.

Coordinate Systems

Four primary right-handed coordinate systems are used for developing the governing equations of motion.

1. A inertial coordinate system is defined with axes x forward, y to port, and z

[1]Note label 'F' is not used to avoid ambiguity with an applied force.

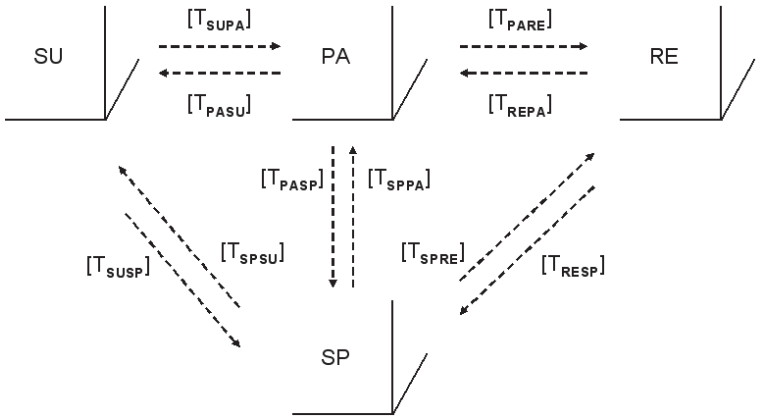

Figure 21: *Primary coordinate systems and transformations used in developing the mathematical model.*

vertically up. This inertial frame can be considered either a fixed-earth coordinate system or a translating-earth coordinate system. This choice does not affect the governing dynamic equations or model implementation described in this section with one minor exception. Viscous damping forces acting on the payload body are calculated based on the payload velocity relative to the inertial coordinate system. As a result, a larger longitudinal damping force will result when using fixed-earth coordinates (as is the case with the current federation implementation) compared with a translating-earth implementation. The script 'SP' is used to refer to the inertial frame (sometimes referred to as the *sp*ace frame).

2. A supply ship frame is attached to the supply ship at the centre of mass with x forward, y to port, and z vertically up. The script 'SU' is used to refer to the supply ship frame.

3. A receiving ship frame that is analogous to the supply ship frame is attached to the receiving ship at its centre of mass. The script 'RE' is used to refer to the receiving ship frame.

4. A payload frame is attached at the centre of mass of the payload with x forward, y to the left, and z up. The script 'PA' is used to refer to the payload frame.

Figure 21 shows the four primary coordinate systems that are used to derive the governing equations and identifies the transformation matrices used to convert between the various coordinate systems. For example, premultiplying a vector by the rotational transformation matrix $[T_{SUPA}]$ transforms it from the supply ship frame (SU) to the payload frame (PA).

Rotation Sequences

The complete RAS equipment model was developed based on the XYZ Euler angle rotation sequence[2] and all internal computations are performed using this convention. SHIPMO3D software uses the ZYX rotation sequence[3]. As a result, input ship motion can be based on this convention and for consistency payload orientation is optionally output using this convention. With both sequences of rotations, the singularity occurs in the unlikely event that the Y rotation (pitch) reaches 90 degrees. Basic equations corresponding to both sets of Euler angles are provided next followed by the procedure for converting between conventions.

In the XYZ convention, three consecutive rotations through angles θ_x, θ_y, then θ_z are performed rotating the global (g) coordinate system into the body-fixed local coordinate system (l). A vector expressed in the global frame $\vec{R}_g$ can be transformed to an equivalent vector expressed in the local frame $\vec{R}_l$ using the rotational transformation matrix from the global to the local coordinate system defined by the XYZ Euler angles

$$\vec{R}_l = \left[T_{g\to l}^{XYZ}\right] \vec{R}_g \tag{1}$$

where

$$\left[T_{g\to l}^{XYZ}\right] = \begin{bmatrix} \mathrm{c}\,\theta_y\,\mathrm{c}\,\theta_z & \mathrm{c}\,\theta_x\,\mathrm{s}\,\theta_z + \mathrm{s}\,\theta_x\,\mathrm{s}\,\theta_y\,\mathrm{c}\,\theta_z & \mathrm{s}\,\theta_x\,\mathrm{s}\,\theta_z - \mathrm{c}\,\theta_x\,\mathrm{s}\,\theta_y\,\mathrm{c}\,\theta_z \\ -\mathrm{c}\,\theta_y\,\mathrm{s}\,\theta_z & \mathrm{c}\,\theta_x\,\mathrm{c}\,\theta_z - \mathrm{s}\,\theta_x\,\mathrm{s}\,\theta_y\,\mathrm{s}\,\theta_z & \mathrm{s}\,\theta_x\,\mathrm{c}\,\theta_z + \mathrm{c}\,\theta_x\,\mathrm{s}\,\theta_y\,\mathrm{s}\,\theta_z \\ \mathrm{s}\,\theta_y & -\mathrm{s}\,\theta_x\,\mathrm{c}\,\theta_y & \mathrm{c}\,\theta_x\,\mathrm{c}\,\theta_y \end{bmatrix} \tag{2}$$

and c and s are abbreviations for the trigonometric cos and sin functions respectively.

Similarly, vectors can be transformed from the local to the global coordinate system using

$$\vec{R}_g = \left[T_{l\to g}^{XYZ}\right] \vec{R}_l \tag{3}$$

where

$$\left[T_{l\to g}^{XYZ}\right] = \left[T_{g\to l}^{XYZ}\right]^{-1} = \left[T_{g\to l}^{XYZ}\right]^{T} \tag{4}$$

The angular velocity of the body, expressed in local coordinates, can be related to the time derivatives of the XYZ Euler angles $(\dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z)$ and the Euler angles themselves such that

$$\left\{ \begin{array}{c} \omega_{xl} \\ \omega_{yl} \\ \omega_{zl} \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} \dot{\theta}_x \cos\theta_y \cos\theta_z + \dot{\theta}_y \sin\theta_z \\ -\dot{\theta}_x \cos\theta_y \sin\theta_z + \dot{\theta}_y \cos\theta_z \\ \dot{\theta}_x \sin\theta_y + \dot{\theta}_z \end{array} \right\} \tag{5}$$

[2]This set of angles is also often referred to as Bryant angles.

[3]This set of angles is referred to as the standard NASA aircraft convention and is commonly used for ship motion analysis.

Alternatively, the angular velocity of the body can be expressed in the global coordinate system as

$$\left\{ \begin{array}{c} \omega_{xg} \\ \omega_{yg} \\ \omega_{zg} \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} \dot{\theta}_x + \dot{\theta}_z \ \sin\theta_y \\ \dot{\theta}_y \ \cos\theta_x - \dot{\theta}_z \ \sin\theta_x \ \cos\theta_y \\ \dot{\theta}_y \ \sin\theta_x + \dot{\theta}_z \ \cos\theta_x \ \cos\theta_y \end{array} \right\} \tag{6}$$

In solving forward dynamics problems using Newton-Euler equations, it is common for the angular velocities in the local coordinate system to be known. These are obtained from numerical integration of angular accelerations expressed in local coordinates determined using Euler's equation or the more general dynamic moment balance. From these it is necessary to extract the time derivatives of the Euler angles. These are required so that they can be numerically integrated to obtain the Euler angles at the next time step. The time derivatives of the Euler angles can be evaluated from the local angular velocities and the Euler angles themselves as

$$\left\{ \begin{array}{c} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} (\cos\theta_z \ \omega_{xl} - \sin\theta_z \ \omega_{yl}) \,/\, \cos\theta_y \\ \sin\theta_z \ \omega_{xl} + \cos\theta_z \ \omega_{yl} \\ (\sin\theta_z \ \omega_{yl} - \cos\theta_z \ \omega_{xl}) \tan\theta_y + \omega_{zl} \end{array} \right\} \tag{7}$$

Differentiating Equation 5 with respect to time results in the local angular acceleration expression $\vec{\alpha}_l^{XYZ}$ given by

$$\left\{ \begin{array}{c} \alpha_x \\ \alpha_y \\ \alpha_z \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} \ddot{\theta}_x \mathrm{c}\theta_y \mathrm{c}\theta_z - \dot{\theta}_x\dot{\theta}_y \mathrm{s}\theta_y \mathrm{c}\theta_z - \dot{\theta}_x\dot{\theta}_z \mathrm{c}\theta_y \mathrm{s}\theta_z + \ddot{\theta}_y \mathrm{s}\theta_z + \dot{\theta}_y\dot{\theta}_z \mathrm{c}\theta_z \\ -\ddot{\theta}_x \mathrm{c}\theta_y \mathrm{s}\theta_z + \dot{\theta}_x\dot{\theta}_y \mathrm{s}\theta_y \mathrm{s}\theta_z - \dot{\theta}_x\dot{\theta}_z \mathrm{c}\theta_y \mathrm{c}\theta_z + \ddot{\theta}_y \mathrm{c}\theta_z - \dot{\theta}_y\dot{\theta}_z \mathrm{s}\theta_z \\ \ddot{\theta}_x \mathrm{s}\theta_y + \dot{\theta}_x\dot{\theta}_y \mathrm{c}\theta_y + \ddot{\theta}_x \end{array} \right\} \tag{8}$$

The previous mathematical relations are used in the subsequent development of the rotational dynamic equations for the RAS equipment.

In the case of the ZYX rotation sequence, consecutive rotations are performed through angles Θ_z, Θ_y, then Θ_x respectively in rotating the global frame into the local frame. The corresponding rotational transformation matrix is

$$\left[T_{g\to l}^{ZYX}\right] = \left[\begin{array}{ccc} \mathrm{c}\Theta_y\mathrm{c}\Theta_z & \mathrm{c}\Theta_y\mathrm{s}\Theta_z & -\mathrm{s}\Theta_y \\ \mathrm{s}\Theta_x\mathrm{s}\Theta_y\mathrm{c}\Theta_z - \mathrm{c}\Theta_x\mathrm{s}\Theta_z & \mathrm{s}\Theta_x\mathrm{s}\Theta_y\mathrm{s}\Theta_z + \mathrm{c}\Theta_x\mathrm{c}\Theta_z & \mathrm{s}\Theta_x\mathrm{c}\Theta_y \\ \mathrm{c}\Theta_x\mathrm{s}\Theta_y\mathrm{c}\Theta_z + \mathrm{s}\Theta_x\mathrm{s}\Theta_z & \mathrm{c}\Theta_x\mathrm{s}\Theta_y\mathrm{s}\Theta_z - \mathrm{s}\Theta_x\mathrm{c}\Theta_z & \mathrm{c}\Theta_x\mathrm{c}\Theta_y \end{array} \right] \tag{9}$$

The local angular velocity vector $\vec{\Omega}^{ZYX}$ is given by

$$\left\{ \begin{array}{c} \Omega_x \\ \Omega_y \\ \Omega_z \end{array} \right\}^{ZYX} = \left\{ \begin{array}{c} -\dot{\Theta}_z\mathrm{s}\Theta_y + \dot{\Theta}_x \\ \dot{\Theta}_z\mathrm{c}\Theta_y\mathrm{s}\Theta_x + \dot{\Theta}_y\mathrm{c}\Theta_x \\ \dot{\Theta}_z\mathrm{c}\Theta_y\mathrm{c}\Theta_x - \dot{\Theta}_y\mathrm{s}\Theta_x \end{array} \right\} \tag{10}$$

The local angular acceleration vector $\vec{\dot{\Omega}}^{ZYX}$ is given by

$$\left\{ \begin{array}{c} \dot{\Omega}_x \\ \dot{\Omega}_y \\ \dot{\Omega}_z \end{array} \right\}^{ZYX} =$$

$$\left\{ \begin{array}{c} -\ddot{\Theta}_z \mathrm{s}\Theta_y - \dot{\Theta}_y \dot{\Theta}_z \mathrm{c}\Theta_y + \ddot{\Theta}_x \\ \ddot{\Theta}_z \mathrm{c}\Theta_y \mathrm{s}\Theta_x - \dot{\Theta}_y \dot{\Theta}_z \mathrm{s}\Theta_y \mathrm{s}\Theta_x + \dot{\Theta}_x \dot{\Theta}_z \mathrm{c}\Theta_y \mathrm{c}\Theta_x + \ddot{\Theta}_y \mathrm{c}\Theta_x - \dot{\Theta}_x \dot{\Theta}_y \mathrm{s}\Theta_x \\ \ddot{\Theta}_z \mathrm{c}\Theta_y \mathrm{c}\Theta_x - \dot{\Theta}_y \dot{\Theta}_z \mathrm{s}\Theta_y \mathrm{c}\Theta_x - \dot{\Theta}_x \dot{\Theta}_z \mathrm{c}\Theta_y \mathrm{s}\Theta_x - \ddot{\Theta}_y \mathrm{s}\Theta_x + \dot{\Theta}_x \dot{\Theta}_y \mathrm{c}\Theta_x \end{array} \right\} \tag{11}$$

Converting between Rotation Sequences

The conversion between Euler rotation sequences is accomplished based on the required equivalency of the rotational transformation matrix, local angular velocity vector, and local angular acceleration vector irrespective of the Euler angle rotation sequence used for evaluating them. As an example, consider the conversion of the ZYX Euler angles and their first and second time derivatives to the XYZ convention. These are the calculations performed on incoming SHIPMO3D ship motions for both the supply and receiving ships.

1. The rotational transformation matrix, local frame angular velocity vector, and local frame angular acceleration vector are evaluated from the ZYX convention Euler angles and their first and second time derivatives using Equations 9, 10, and 11 respectively.

2. The XYZ convention Euler angles are identified by comparing entries in the ZYX and XYZ expressions for the rotational transformation matrices (Equations 9 and 2 respectively) resulting in

$$\left\{ \begin{array}{c} \theta_x \\ \theta_y \\ \theta_z \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} \mathrm{atan}(T_{g\to l}^{ZYX}(3,2)/T_{g\to l}^{ZYX}(3,3)) \\ \mathrm{asin}(T_{g\to l}^{ZYX}(3,1)) \\ \mathrm{atan}(-T_{g\to l}^{ZYX}(2,1)/T_{g\to l}^{ZYX}(1,1)) \end{array} \right\} \tag{12}$$

   It should be noted that quadrant-specific arctangent functions should be used for evaluating the above equation.

3. The time derivatives of the Euler angles are obtained by evaluating Equation 7 using the identified XYZ Euler angles obtained in Step 2 and the local angular velocity vector evaluated in Step 1 noting that $\vec{\Omega}^{ZYX} = \vec{\omega}^{XYZ}$ resulting in

$$\left\{ \begin{array}{c} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{array} \right\}^{XYZ} = \left\{ \begin{array}{c} (\mathrm{c}\theta_z \Omega_x - \mathrm{s}\theta_z \Omega_y)/\mathrm{c}\theta_y \\ \mathrm{s}\theta_z \Omega_x + \mathrm{c}\theta_z \Omega_y \\ (\mathrm{s}\theta_z \Omega_y - \mathrm{c}\theta_z \Omega_x)\tan\theta_y + \Omega_z \end{array} \right\} \tag{13}$$

4. The second time derivatives of the Euler angles are obtained by solving Equation 8 for the unknown second time derivatives of the XYZ Euler angles noting that $\vec{\Omega}^{ZYX} = \vec{\alpha}^{XYZ}$. Performing this analytically results in the expression

$$\left\{ \begin{array}{c} \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{array} \right\}^{XYZ} = \left[\begin{array}{ccc} \mathrm{c}\theta_z/\mathrm{c}\theta_y & -\mathrm{s}\theta_z/\mathrm{c}\theta_y & 0 \\ \mathrm{s}\theta_z & \mathrm{c}\theta_z & 0 \\ -\mathrm{c}\theta_z\mathrm{s}\theta_y/\mathrm{c}\theta_y & \mathrm{s}\theta_z\mathrm{s}\theta_y/\mathrm{c}\theta_y & 1 \end{array} \right]$$

$$\left\{ \begin{array}{c} \dot{\Omega}_x + \dot{\theta}_x\dot{\theta}_y\mathrm{s}\theta_y\mathrm{c}\theta_z + \dot{\theta}_x\dot{\theta}_z\mathrm{c}\theta_y\mathrm{s}\theta_z - \dot{\theta}_y\dot{\theta}_z\mathrm{c}\theta_z \\ \dot{\Omega}_y - \dot{\theta}_x\dot{\theta}_y\mathrm{s}\theta_y\mathrm{s}\theta_z + \dot{\theta}_x\dot{\theta}_z\mathrm{c}\theta_y\mathrm{c}\theta_z + \dot{\theta}_y\dot{\theta}_z\mathrm{s}\theta_z \\ \dot{\Omega}_z - \dot{\theta}_x\dot{\theta}_y\mathrm{c}\theta_y \end{array} \right\} \quad (14)$$

Conversion from the XYZ convention to the ZYX convention as is optionally performed on the output payload rotational kinematics is formulated analogously.

The computational routines that have been implemented to perform these conversions (CONVZYXTOXYZ and CONVXYZTOZYX) include a series of calculations to confirm correctness of the conversion at run time to within a pre-set tolerance (nominally set to 10^{-6} though confirmed to 10^{-12}). This check involves evaluating the appropriate transformation matrices, local angular velocity vectors, and local angular acceleration vectors using both sets of Euler angles and corresponding time derivatives and ensuring that no corresponding pair of matrix or vector elements exceeds the tolerance. If an element were to exceed the tolerance the simulation would stop and identify the source of the error. Conversion errors are not expected and have not been observed.

Interpolated Highline Points

Coordinates of a user-selected number of points along the highline are required to support visualization, particularly if cable sag and dynamics are subsequently included in the model. These points are currently obtained by interpolating a user-selected number of approximately equally-spaced points along the highline. The points include the supply ship attachment point (Point A), receiving ship attachment point (Point H), and payload/highline interface point (Point K). The number of segments on either side of the highline are apportioned based on the relative cable spans and points on each side of the payload interface point are equally spaced. It is important to note that these output points are re-interpolated at each output time step and correspondingly reapportioned on both sides of the payload as appropriate. The result is that the derivatives of the interpolated highline points are not continuous. Subsequently, if the points are made to correspond to elements of cable mass, the formulation would inherently ensure that the derivatives are continuous. It would also be readily possible to output the velocity and acceleration of individual point masses.

3.2.2 Ship Kinematics

Dead Reckoning
The supply and receiving ship motions, evaluated by SHIPMO3D, are input to the RAS gear model as the instantaneous values of ship centre of mass linear displacement, velocity, and accelerations; and the Euler angles as well as their first and second time derivatives.

Due to the presence of active control systems in the RAS gear model and the potentially long cycle time of the federation relative to high-frequency control that may be required by the RAS equipment model, a ship motion dead reckoning algorithm has been implemented that allows the RAS gear model to use a smaller integration time step than the federation cycle time step. As a result, the RAS gear dynamics can be evaluated a user-selected number of times per federation time step. Each of the input ship motion components is dead-reckoned on the assumption of constant ship acceleration components A_i over the federation cycle time such that the 12 velocity components V_i (6 translational and 6 rotational) are extrapolated using

$$V_i = V_{0i} + A_{0i}\Delta t \tag{15}$$

where subscript i indicates the specific motion component, subscript 0 indicates the value at the beginning of the federation time step, and Δt is the elapsed time since the beginning of the federation time step when the ship accelerations were evaluated. The dead-reckoned displacement components are similarly evaluated as

$$D_i = D_{0i} + V_{0i}\Delta t + \frac{1}{2}A_{0i}\Delta t^2 \tag{16}$$

Kinematics of Points of Interest
The linear displacement and linear velocity of points of interest, namely cable attachment points on the ships and payload body, must be evaluated from the basic ship and payload kinematic variables. The global positions $\vec{r}_g$ of points of interest are evaluated from

$$\vec{r}_g = \vec{r}_{cm} + [T_{l\to g}]\,\vec{r}_l \tag{17}$$

where $\vec{r}_{cm}$ is the global position vector to the centre of mass of the body to which the point of interest is attached, $[T_{l\to g}]$ is the rotational transformation applicable to the body, and $\vec{r}_l$ is the relative position vector from the body centre of mass to the point of interest expressed in the body-fixed coordinate system (SU, RE, or PA as appropriate).

The linear velocities of points of interest $\vec{v}_g$ are evaluated from

$$\vec{v}_g = \vec{v}_{cm} + [T_{l\to g}]\,(\vec{\omega}_l \times \vec{r}_l) \tag{18}$$

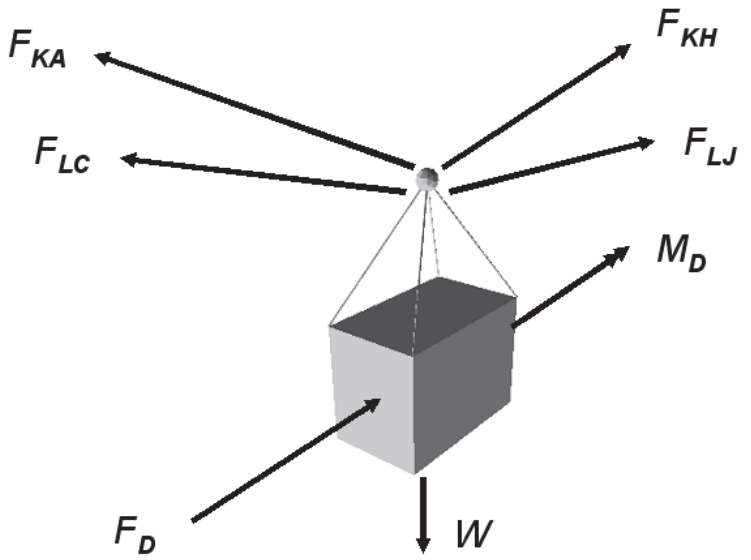

Figure 22: *Payload body.*

where $\vec{v}_{cm}$ is the linear velocity of the centre of mass of the attachment body and $\vec{\omega}_l$ is the angular velocity of the attachment body expressed in the local body-fixed coordinate system (SU, RE, or PA as appropriate).

It should be noted that the RAS governing dynamic equations do not depend explicitly on the acceleration of the attachment points.

3.2.3 Payload Dynamics

Overview

The payload dynamic model consists of a 6 degree-of-freedom free rigid body representation of the payload body and its harness. Forces acting on the body consist of the highline tension components directed both toward the supply and receiving ships, in-/out-haul system tensions, weight, and a set of generalized viscous damping forces and moments that provide a means of including energy dissipation in the model. The equations of motion are written for and about the body centre of mass. The state variables used for the payload body model are the linear position and velocity of the body centre of mass, orientation specified by the XYZ Euler angle sequence, and orientation time derivatives specified by the body-fixed angular velocity vector.

Figure 22 illustrates a payload body with applicable forces that will be subsequently defined and evaluated.

Translational Equations of Motion

The displacement vector $\vec{r}_P$ from the origin of the inertial frame to the payload centre of mass is

$$\vec{r}_P = r_x\vec{\imath} + r_y\vec{\jmath} + r_z\vec{k} \tag{19}$$

The linear velocity of the payload centre of mass $\vec{v}_P$ is defined as the time derivative of the displacement vector

$$\vec{v}_P = v_x\vec{\imath} + v_y\vec{\jmath} + v_z\vec{k} \tag{20}$$

Similarly, the linear acceleration of the payload centre of mass $\vec{a}_P$ is the derivative of the linear velocity

$$\vec{a}_P = a_x\vec{\imath} + a_y\vec{\jmath} + a_z\vec{k} \tag{21}$$

Newton's law is used to relate the equipollent forces[4] $\vec{F}_P$ acting on the payload to the linear accelerations of the payload centre of mass expressed in the inertial coordinate system (SP)

$$\sum \vec{F}_P = m\ \vec{a}_P \tag{22}$$

where m is the mass of the payload.

Equation 22 is solved for the linear accelerations. These are numerically integrated twice resulting in the linear velocities and displacements of the payload expressed in global coordinates.

Rotational Equations of Motion

Euler's equation is used to solve for the absolute angular accelerations of the payload expressed in the payload frame $\dot{\vec{\omega}}$ from the vector of equipollent moments acting on the payload and the angular velocities of the payload [11]

$$[I]\left\{\dot{\omega}_P^l\right\} = \{M_P\} + \{M_{gyr}\} \tag{23}$$

Matrix $[I]$ is the inertia matrix for the payload

$$[I] = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{24}$$

Vector $\left\{\dot{\omega}^l\right\}$ is the vector of angular accelerations expressed in the payload coordinate system (PA)

$$\left\{\dot{\omega}_P^l\right\} = \left\{ \begin{array}{c} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{array} \right\}_P \tag{25}$$

[4]Equipollent forces and moments constitute a set of forces acting at the centre of mass of a rigid body and moments about the centre of mass that are equivalent to the combined effect of all forces and moments acting on the body.

Vector $\{M_P\}$ is the equipollent moment vector

$$\{M_P\} = \left\{ \begin{array}{c} M_x \\ M_y \\ M_z \end{array} \right\}_P \tag{26}$$

Vector $\{M_{gyr}\}$ is the vector of gyroscopic moments

$$\{M_{gyr}\} = \left\{ \begin{array}{c} -\omega_y\omega_z\left(I_{zz} - I_{yy}\right) - \omega_y\omega_x I_{zx} - \left(\omega_y^2 - \omega_z^2\right) I_{zy} + \omega_x\omega_z I_{yx} \\ -\omega_x\omega_z\left(I_{xx} - I_{zz}\right) - \omega_y\omega_z I_{xz} - \left(\omega_z^2 - \omega_x^2\right) I_{xz} + \omega_y\omega_x I_{zy} \\ -\omega_x\omega_y\left(I_{yy} - I_{xx}\right) - \omega_x\omega_z I_{yz} - \left(\omega_x^2 - \omega_y^2\right) I_{yx} + \omega_y\omega_z I_{xz} \end{array} \right\} \tag{27}$$

where it is recognized that all local angular velocity components ω_i apply to the payload body.

The angular accelerations of the payload in the local coordinate system are numerically integrated to obtain angular velocities in the local coordinate system. From these, Equation 7 is used to determine the time derivatives of the Euler angles which are numerically integrated to obtain the Euler angles for the next time step.

Forces and Moments

Tensions in the highline cable, in-haul cable, and out-haul cable are designated T_{hl}, T_{ih}, and T_{oh} respectively. Cable forces acting on the payload body are obtained by projecting cable tensions onto unit vectors directed from the payload attachment points to the ship attachment points such that

$$\vec{F}_{KA} = T_{hl}\frac{\vec{r}_{KA}}{L_{KA}} \tag{28}$$

$$\vec{F}_{KH} = T_{hl}\frac{\vec{r}_{KH}}{L_{KH}} \tag{29}$$

$$\vec{F}_{LC} = T_{ih}\frac{\vec{r}_{LC}}{L_{LC}} \tag{30}$$

$$\vec{F}_{LJ} = T_{oh}\frac{\vec{r}_{LJ}}{L_{LJ}} \tag{31}$$

where the first subscript on forces indicates the point at which the force is applied and the second subscript indicates the other cable attachment point along the line of action of the force[5]. Using these force definitions, the equipollent force acting on the payload body, expressed in the inertial coordinate system (SP), is given by

$$\vec{F}_P = \vec{F}_{KA} + \vec{F}_{KH} + \vec{F}_{LC} + \vec{F}_{LJ} + \vec{F}_D + \vec{W} \tag{32}$$

[5]While somewhat unconventional, this notation concisely identifies forces present in the RAS equipment model.

where $\vec{F}_D$ is an effective viscous damping force applied directly to the payload body centre of mass (P) and $\vec{W}$ is the body weight vector having magnitude mg directed downward in the inertial frame where m is the payload body mass and g is the acceleration due to gravity.

The equipollent moment vector, expressed in the payload local coordinate system (PA), is given by

$$\vec{M}_P = \vec{r}_{PK} \times [T_{SPPA}] \left(\vec{F}_{KA} + \vec{F}_{KH} \right) + \vec{r}_{PL} \times [T_{SPPA}] \left(\vec{F}_{LC} + \vec{F}_{LJ} \right) + \vec{M}_D \qquad (33)$$

where $\vec{M}_D$ is a viscous damping moment applied directly to the payload body.

The viscous damping force vector is defined as

$$\vec{F}_D = - \left\{ \begin{array}{l} C_{tx}\ v_{Px} \\ C_{ty}\ v_{Py} \\ C_{tz}\ v_{Pz} \end{array} \right\} \qquad (34)$$

and the viscous damping moment is defined as

$$\vec{M}_D = - \left\{ \begin{array}{l} C_{rx}\ \omega^l_{Px} \\ C_{ry}\ \omega^l_{Py} \\ C_{rz}\ \omega^l_{Pz} \end{array} \right\} \qquad (35)$$

where C_{ij} are viscous damping coefficients with subscript i indicating translational (t) or rotational (r) motion and subscript j indicating the component direction. A viscous damping term is included to provide a mechanism to represent energy dissipation associated with aerodynamic and friction effects on the payload motions.

3.2.4 Highline Equipment

The ram tensioner and the highline winch model that controls highline tension was formulated based on using the 'active cable length' defined as the total cable length outboard of the highline winch as the primary degree of freedom. This results in a compact physically-meaningful model for which the parameters can largely be derived from relatively high-level design parameters available in the project reference documents. The ram tensioner model supports the following behaviours:

- ram compression consistent with multiple cable falls;
- deadband limits and centering tolerance;
- proportional control of the ram operating point;
- integral control for ship drift;

- winch speed and power limits;
- winch spooling beyond maximum tension;
- cable stretch;
- cable detensioning due to approaching the ram extension limit; and
- tension differences due to friction.

The following provides the mathematical basis for the model and its implementation.

1. The model must first be initialized to the user-selected operating parameters and initial conditions. First the nominal ram operating cable tension $T_{hl\,0}$ is evaluated to be

$$T_{hl\,0} = \frac{P_0 A}{NF} \tag{36}$$

where P_0 is the nominal system gas pressure, A is the ram piston area evaluated from the user input piston diameter D, and NF is the number of cable falls in the ram tensioner.

The nominal tension is used to obtain an estimate of the peak cable friction ΔT_f as

$$\Delta T_f = \gamma T_{hl\,0} \tag{37}$$

where γ is the ram tensioner friction factor.

Next, the ram compression initial condition is used to determine the actual initial operating state of the ram. First, the change is system gas volume ΔV is evaluated from the initial ram compression as

$$\Delta V = A\Delta_{ram\,0} \tag{38}$$

where Δ_{ram} is the ram compression and subscript 0 implies the initial condition value.

The actual system operating volume V is then

$$V = V_0 - \Delta V \tag{39}$$

where V is the total gas volume of the ram, accumulator, and associated plumbing and subscript 0 implies the nominal volume when the ram cylinder is centred.

The corresponding current gas pressure P in the system is determined from the ideal gas law such that

$$P = \frac{P_0 V_0}{V} \tag{40}$$

The initial highline tension (excluding frictional effects) is

$$T_{hl} = \frac{PA}{NF} \tag{41}$$

The above calculations are performed once to initialize the ram tensioner model. Subsequent steps are repeated each time the dynamic equation derivatives are required.

2. The 'active cable length' q_c defined as the total length of cable outboard of the highline winch (excluding cable stretch) and is initialized to be

$$q_c = \left(L_{AK} + L_{HK} + L_{AD} + L_{DF} + L_{DE0} - NF\Delta_{ram0}\right) / \left(1 + \frac{T_{hl}}{A_c E_c}\right) \tag{42}$$

where L_{AD} refers to the constant cable length between the point at which the highline cable first becomes inboard on the supply ship and the base of the ram tensioner (D), L_{DG} is the constant cable length from the base of the ram tensioner (D) to the highline winch (G), and $L_{DE\,0}$ is the initial length of cable stored in the ram tensioner when compressed to its centred/equilibrium position. This length remains constant except for winch activity.

3. The length of cable stretch $L_{stretch}$ in the highline cable, accounting for all cable between the receiving ship and the highline winch drum, is calculated as

$$L_{stretch} = \frac{T_{hl} q_c}{A_c E_c} \tag{43}$$

where A_c is the effective cable cross-section and E_c is the modulus of elasticity of the highline cable. In addition to initialization, cable stretch is updated between internal integration time steps. This is necessary as the cable stretch is not declared as an explicit state variable as its time derivative is not available.

4. Ship and payload kinematics alter L_{AK} and L_{HK}. As a result, Equation 42 can be solved for the instantaneous ram compression Δ_{ram} making use of the definition for $L_{stretch}$ from Equation 43 resulting in

$$\Delta_{ram} = \left(L_{AK} + L_{HK} + L_{AD} + L_{DF} + L_{DE0} - L_{stretch} - q_c\right) / NF$$

5. The highline tension is evaluated as appropriate depending on the amount of ram compression. While operating within its normal bounds, the tension is evaluated as

$$T_{hl} = \frac{P_0 V_0 A}{NF\ V_0 - A\Delta_{ram}} \tag{44}$$

If the ram becomes extended to the point that it begins compressing the extension bumper at $\Delta_{ram} = L_{spring}$ then the cable becomes partially detensioned as the bumper spring is compressed. In this case for the range $L_{spring} < \Delta_{ram} \leq L_{stop}$

$$T_{hl} = \frac{P_0 V_0 A}{NF\ V_0 - A\Delta_{ram}} - \frac{k_{bump}\left(\Delta_{ram} - L_{spring}\right)}{NF} \tag{45}$$

If the ram becomes fully extended such that its hard stop is reached at L_{stop}, the cable is detensioned fully such that $T_{hl} = 0$ and the ram extension is bounded not to exceed L_{stop}.

6. The highline winch is enabled when the ram compression or extension exceeds a pre-set deadband limit Δ_{db}. The winch then remains active until the ram is recentred within a pre-set tolerance Δ_{tol}.

   If the winch is active, the highline cable length changes at a rate established by the proportional highline winch controller, but bounded by an upper speed limit and available winch power. Additionally, if integral drift control is enabled, a component of winch speed is determined by the integral error on ram compression such that

   $$\dot{q}_c = c_{winch}\Delta_{ram} + k_i \int \Delta_{ram} \mathrm{d}t \tag{46}$$

   where the first term is included when the winch is active due to ram position and the second term is included when integral drift control is enabled by the user by setting a nonzero value for k_i.

   The maximum highline winch payout rate is limited to the lower of the maximum winch cable design speed $\dot{q}_{c\ max\ design}$ and the maximum winch power speed $\dot{q}_{c\ max\ power}$ where

   $$\dot{q}_{c\ max\ power} = \frac{P_{hl\ winch\ max}}{T_{hl}} \tag{47}$$

   where $P_{hl\ winch\ max}$ is the peak power rating of the winch.

7. If the maximum allowable cable tension is exceeded, cable will be pulled off the winch drum. The rate at which this occurs can be related to the rate at which the length of cable outboard of the supply ship is changing relative to the rate at which the cable length inboard of the supply ship is changing. This assumes that the ram tensioner is at its travel limit when the maximum tension is reached such that the time derivative of the cable length stored in the ram tensioner is zero. This assumption is likely to be valid for physically-meaningful system design parameters. Removal of this assumption would require a detailed dynamic model of the ram tensioner with an associated degree of freedom and

the requirement to determine low-level (and difficult to evaluate) design parameters of the ram tensioner system.

In this case, the highline tension is set to its maximum value of $T_{hl\,max}$ and the derivative of the active cable length is set to

$$\dot{q}_c = \max\left(\dot{L}_{HK} + \dot{L}_{AK}, \dot{q}_c\right) \tag{48}$$

8. The tension of the highline cable outboard of the kingpost is affected by frictional losses in the ram tensioner. Therefore, the highline tension must be corrected for the frictional resistance. A sinusoidal smoothing function is used to avoid discontinuity in the friction force as the direction of cable movement through the ram tensioner reverses. The frictional contribution to highline tension is therefore added to the highline tension as

$$T_{hl} = T_{hl} + \Delta T_f \sin\left(\frac{\pi\left(\dot{L}_{HK} + \dot{L}_{AK}\right)}{2T_{f\ threshold}}\right) \tag{49}$$

when the cable speed is less then V_{thresh} and

$$T_{hl} = T_{hl} + \Delta T_f \tag{50}$$

otherwise. The frictional component opposes the direction of cable motion through the ram tensioner system subject to the constraint that the frictional component cannot cause the highline tension to become negative.

In the RAS gear simulation the ram tensioner model serves the purpose of evaluating the highline tension, the derivative of the active cable length ($\dot{q}_c$), and the derivative of the ram centring position error $\dot{q}_{error}$ that is simply recognized to be the ram compression Δ_{ram}.

3.2.5 Traversing System

The in-haul/out-haul system model comprises both the physical behaviour of the system and representation of the input of the human operator. Effectively, the model tracks a commanded traverse velocity profile that is specified in the input file as a function of progress from the supply ship towards the receiving ship. Position is measured by the ratio of the projection of the payload/highline attachment point (Point K) onto the straight-line vector joining the two highline attachment points (Points A and H) to the distance between highline attachment points (Points A and H). Depending on whether the commanded velocity is away from or towards the supply ship, the in-haul and out-haul winches are placed in tension control mode or velocity control mode as appropriate. The winch causing tension in the direction of

commanded travel is the one that is in velocity control mode. Tension control mode consists of applying a constant opposing tension subject to constraints on maximum winch payout rate and up-take speed and available winch power. Velocity control mode implements a proportional controller between the error between actual and commanded velocities and differential cable tension. It is subjected to the same constraints as the tension control mode with an additional constraint on the peak tension that can be developed beyond which the winch spools cable. This results in a compact physically-meaningful model for which the parameters can largely be derived from information available in the project reference documents.

The traversing system model implements the following procedure.

1. Project the payload relative position and relative velocity onto the line connecting the highline attachment points to determine the effective ratio of payload position relative to the distance between highline attachment points $r_{payload}$ and the effective traversing speed $v_{payload}$. The corresponding values are calculated as

$$r_{payload} = \frac{L_{AK}\left(\frac{\vec{r}_{AK}}{L_{AK}} \cdot \frac{\vec{r}_{AH}}{L_{AH}}\right)}{L_{AH}} \tag{51}$$

and

$$v_{payload} = \dot{L}_{AK}\left(\frac{\vec{r}_{AK}}{L_{AK}} \cdot \frac{\vec{r}_{AH}}{L_{AH}}\right) \tag{52}$$

2. Determine the rates at which the in-haul and out-haul cable lengths, $V_{in-haul}$ and $V_{out-haul}$ respectively, are changing

$$V_{in-haul} = \dot{L}_{CL} \tag{53}$$
$$V_{out-haul} = \dot{L}_{BI} + \dot{L}_{JL} \tag{54}$$

3. Interpolate the commanded traverse velocity from the user-provided tabulated control points.

4. Set the traverse winch control modes such that the winch causing motion is in velocity control mode and the winch opposing the commanded velocity is in tension control mode. The control modes are implemented such that for tension control:

$$T = T_{pretension} \tag{55}$$

and for velocity control:

$$T = G(V_{cmd} - v_{payload}) + T_{pretension} \tag{56}$$

where G is the velocity controller gain.

5. Limit tensions based on:

- peak allowable cable tension that is specified separately for the in-haul and out-haul winches as $T_{ih\ max}$ and $T_{oh\ max}$ respectively;
- maximum winch speed such that cables are completely detensioned if the required winch take-up speed exceeds the maximum achievable winch speed $v_{tu\ ioh\ max}$; and
- available winch power where the speed limits are evaluated analogously to the maximum highline speed based on power as

$$V_{power\ ih\ max} = \frac{P_{ih\ max}}{T_{ih}} \tag{57}$$

$$V_{power\ oh\ max} = \frac{P_{oh\ max}}{T_{oh}} \tag{58}$$

$$\tag{59}$$

  for the in-haul and out-haul winches respectively.

The purpose of the traversing system model is to evaluate the in-haul and out-haul cable tensions.

3.2.6 Ship Interface Forces and Moments

The equipollent force vector acting on the supply ship, expressed in the inertial coordinate frame (SP), results from force contributions arising from each of the highline cable, in-haul cable, and out-haul cable. The individual cable forces are obtained, as with the payload body cable forces, by projecting the cable tensions onto the cable force lines of action such that

$$\vec{F}_{AK} = T_{hl}\frac{\vec{r}_{AK}}{L_{AK}} \tag{60}$$

$$\vec{F}_{CL} = T_{ih}\frac{\vec{r}_{CL}}{L_{CL}} \tag{61}$$

$$\vec{F}_{BI} = T_{oh}\frac{\vec{r}_{BI}}{L_{BI}} \tag{62}$$

The equipollent force acting on the supply ship, expressed in the inertial frame (SP), is then

$$\vec{F}_S^{SP} = \vec{F}_{AK} + \vec{F}_{CL} + \vec{F}_{BI} \tag{63}$$

The equipollent moment acting on the supply ship, expressed in the body-fixed supply ship coordinate frame (SU), is obtained by evaluating the moments of the applied cable forces about the supply ship centre of mass such that

$$\vec{M}_S^{SU} = \vec{r}_{SA} \times [T_{SPSU}]\,\vec{F}_{AK} + \vec{r}_{SC} \times [T_{SPSU}]\,\vec{F}_{CL} + \vec{r}_{SB} \times [T_{SPSU}]\,\vec{F}_{BI} \tag{64}$$

Analogous expressions are derived for the receiving ship. The forces acting on the receiving ship are given by

$$\vec{F}_{HK} = T_{hl}\frac{\vec{r}_{HK}}{L_{HK}} \tag{65}$$

$$\vec{F}_{JL} = T_{oh}\frac{\vec{r}_{JL}}{L_{JL}} \tag{66}$$

$$\vec{F}_{IB} = T_{oh}\frac{\vec{r}_{IB}}{L_{IB}} \tag{67}$$

The equipollent force acting on the receiving ship is

$$\vec{F}_R^{SP} = \vec{F}_{HK} + \vec{F}_{JL} + \vec{F}_{IB} \tag{68}$$

The equipollent moment acting on the receiving ship is

$$\vec{M}_R^{RE} = \vec{r}_{RH} \times [T_{SPRE}]\,\vec{F}_{HK} + \vec{r}_{RJ} \times [T_{SPRE}]\,\vec{F}_{JL} + \vec{r}_{RI} \times [T_{SPRE}]\,\vec{F}_{IB} \tag{69}$$

The equipollent forces and moments are evaluated in global and local coordinate systems respectively consistent with common practice dictated by how these forces and moments are subsequently used for propagating the ship motion solution.

3.2.7 Solution Strategy

The equations governing the RAS equipment behaviour and the payload are solved using the conventional approach. The core of the solution is the composite state vector. It contains linear displacements of the payload expressed in the inertial frame, XYZ Euler angles describing the orientation of the payload relative to the inertial frame, linear velocities of the payload expressed in the inertial frame, angular velocities of the payload expressed in the payload frame, the active highline cable length, and the integrated ram position error. The time derivatives of each component of the composite state vector are calculated based on the theory presented in this section. The contents of the state vector and the way in which the derivatives are evaluated are summarized in Table 5. Derivatives are evaluated in a subroutine that is available to the numerical integration algorithm whenever derivatives are required.

The general solution procedure is as follows.

1. Input data is read from the RASEQII.INP input file on the first pass through the RAS equipment model.

2. Parameter values are initialized and initial conditions are established for each of the 14 state variables.

Table 5: *Composite state vector.*

contents	$\vec{q}$	time derivative of $\vec{q}$, $\dot{\vec{q}}$
linear displacements of the payload in the inertial frame	$q(1)$ $q(2)$ $q(3)$	linear velocity of the payload in the inertial frame: $\dot{q}(1) = q(7)$, $\dot{q}(2) = q(8)$, $\dot{q}(3) = q(9)$
XYZ Euler angles from the inertial to the payload frame	$q(4)$ $q(5)$ $q(6)$	calculated using Equation 7 that relates the time derivatives of the XYZ Euler angles to the body-fixed angular velocity vector expressed in the local (payload) frame
linear velocity of the payload in the inertial frame	$q(7)$ $q(8)$ $q(9)$	solved using Newton's law applied to the payload in the inertial frame (Equation 22)
absolute angular velocity of the payload expressed in the payload frame	$q(10)$ $q(11)$ $q(12)$	solved using the general form of Euler's equation applied to the payload in the payload frame (Equation 23)
active highline cable length and integrated ram position error	$q(13)$ $q(14)$	evaluated from the highline cable model where $\dot{q}(13)$ is determined by the rate of cable payout or take-up (when applicable) and $\dot{q}(14) = \Delta_{ram}$

3. The ram tensioner and traversing system models are run to evaluate the three cable tensions (T_{hl}, T_{ih}, and T_{oh}).

4. The dynamic equations governing the payload dynamics are evaluated to obtain the payload linear and angular accelerations.

5. All 14 derivatives are stored in the composite time derivative of the state vector.

6. All 14 differential equations are numerically advanced by one internal time step.

7. The solution returns to Step 3 for the next time step.

Numerical integration of the governing equations of motion is accomplished using a fixed time step fourth-order Runge Kutta numerical integrator. This integrator is appropriate given the relatively low frequency content expected in the dynamic solution and the need to advance the solution efficiently with a predictable number of calculations per time step such that time performance can be estimated reliably and guaranteed. Experience has shown that this integration method performs well for this particular problem.

3.3 Implementation

3.3.1 Software Structure

The mathematical models have been implemented in Fortran. This decision was based on the developer's experience with Fortran and the ability to re-use various proven routines and mathematical functions.

All modelling assumes that input is provided in consistent basic units. This allows the models to be run using either metric or Imperial units; with the only additional requirement being that the correspondingly correct value of acceleration due to gravity (g) be provided in the input file. Basic units are kg, N, m, rad, and s in metric and slug, lbf, ft, rad, and s in Imperial.

Individual subroutines have been extensively documented in the code clearly defining the calculation sequences used, variable definitions and declarations, and subroutine inputs and outputs. The choice to include the extensive documentation directly in the code is to mitigate the possibility of the code and associated documentation diverging over time. The material included in this report is intended to provide a high-level overview of the modelling. Implementation details are available in the source code. Similarly, the definitions of input parameters are clearly defined in the RASEQII.INP input file as well as in this report.

3.3.2 Software Components

The main program source code is contained in one file: RASEQII.FOR. In addition, six include files (*.inc), briefly described below, contain common blocks for passing global variables within the RASEQII software module.

FORCES passes output force information within the RASEQII subroutine.

GEOMETRY stores highline force information such that it is available for animation outside the RASEQII subroutine.

HIGHLINEPOINTS contains information related to the definition of intermediate points along the highline for output.

MOTIONS passes kinematic information within the RASEQII subroutine.

OUTDAT passes internal variables to the main RASEQII subroutine for output.

PARAMS passes physical parameters and initial conditions within the RASEQII subroutine.

The source code is documented extensively to facilitate maintenance and extension of the program. All global variable definitions are provided in the include files where the corresponding common blocks are defined. All local variables are defined at the top of the subroutine where the variables are used. In all cases, the details of calculation sequences can be extracted directly from the comment lines embedded in the code.

The following briefly summarizes calculations performed in each of the 15 subroutines comprising the RASEQII simulation module.

CONVXYZTOZYX converts Euler angles and their first two time derivatives from the XYZ convention to the ZYX convention.

CONVZYXTOXYZ converts Euler angles and their first two time derivatives from the ZYX convention to the XYZ convention.

CROSS forms the cross product $\vec{r} = \vec{p} \times \vec{q}$.

DEFPTS evaluates intermediate highline cable points.

INVERT inverts a 3×3 matrix using the determinant method.

MULT multiplies a 3×3 matrix by a 3 element vector $\{c\} = [A]\{b\}$.

PTMO evaluates the global position and velocity of a body attachment point and the local to global and global to local transformation matrices.

RAMTEN evaluates the highline tension and rate of change of highline cable length as well as the derivative required to evaluate the integral contribution to the highline cable length derivative.

RASEQII uses input ship motions to calculate actively-controlled RAS equipment highline tension and associated equipollent forces (space frame) and equipollent moments (ship frames) acting on both the supply and receiving ships as well as the instantaneous payload position and orientation.

RK4 implements fixed time step fourth order Runge Kutta numerical integration.

SHIPDR implements a constant acceleration based dead-reckoning algorithm for ship motion described by inertial frame linear displacements, velocities, and accelerations; and orientations described by Bryant angles with the first and second time derivatives of orientation being the first and second time derivatives of the Bryant angles.

SMULT multiplies two 3×3 matrices together $[C] = [A][B]$.

TRVTEN evaluates the cable tensions on the in-haul and out- haul cables based on a proportional control law.

UNTVEC evaluates the distance and rate of change of distance between points A and B as well as unit vectors directed from A to B and B to A.

XDOT evaluates derivatives for the RAS equipment model and the equipollent forces and moments acting on the supply and receiving ships as well as RAS cable tensions.

3.3.3 Subroutine Interface

The input and output data passed through the RASEQII subroutine call are described in this section.

The input data consists of the following variables:

p integer flag indicating whether the RAS equipment model should be evaluated at the current input time (p=0) or whether the solution should be propagated forward in time by one federation time step (p=1).

tinp double precision variable containing federation simulation time at the beginning of the current time step.

rsinp(6) double precision vector containing the (x,y,z) coordinates of the supply ship centre of mass relative to the origin of the inertial coordinate system (SP) followed by the Euler angles describing the orientation of the supply ship body-fixed coordinate system (SU) relative to the inertial coordinate system (SP).

vsinp(6) double precision vector containing the (x,y,z) components of the translational velocity of the supply ship centre of mass relative to the origin of the inertial coordinate system (SP) and the first time derivatives of the Euler angles describing the supply ship orientation.

asinp(6) double precision vector containing the (x,y,z) components of the translational acceleration of the supply ship centre of mass relative to the origin of the inertial coordinate system (SP) and the second time derivatives of the Euler angles describing the supply ship orientation.

rrinp(6) double precision vector containing the (x,y,z) coordinates of the receiving ship centre of mass relative to the origin of the inertial coordinate system (SP) followed by the Euler angles describing the orientation of the receiving ship body-fixed coordinate system (RE) relative to the inertial coordinate system (SP).

vrinp(6) double precision vector containing the (x,y,z) components of the translational velocity of the receiving ship centre of mass relative to the origin of the inertial coordinate system (SP) and the first time derivatives of the Euler angles describing the receiving ship orientation.

arinp(6) double precision vector containing the (x,y,z) components of the translational acceleration of the receiving ship centre of mass relative to the origin of the inertial coordinate system (SP) and the second time derivatives of the Euler angles describing the receiving ship orientation.

nhlpntsin integer identifying the number of points for which coordinates along the highline cable should be interpolated and output (the number should range from 3 to 20 inclusive).

The output data consists of the following variables:

feqsspout(3) double precision vector of equipollent force components (x,y,z) acting on the supply ship expressed in the inertial coordinate system (SP).

meqsshout(3) double precision vector of equipollent moment components (x,y,z) acting on the supply ship expressed in the supply ship coordinate system (SU).

feqrspout(3) double precision vector of equipollent force components (x,y,z) acting on the receiving ship expressed in the inertial coordinate system (SP).

meqrshout(3) double precision vector of equipollent moment components (x,y,z) acting on the receiving ship expressed in the receiving ship coordinate system (RE).

rpspout(6) double precision vector containing the coordinates (x,y,z) of the payload body centre of mass relative to the inertial coordinate system expressed in the inertial frame (SP) followed by the Euler angles describing the orientation of the payload body coordinate system (PA) relative to the inertial coordinate system (SP).

vpspout(6) double precision vector containing the translational velocity components (x,y,z) of the payload body centre of mass relative to the inertial coordinate system expressed in the inertial frame (SP) followed by the angular velocity of the payload body expressed in the payload coordinate system (PA)[6].

apspout(6) double precision vector containing the translational acceleration components (x,y,z) of the payload body centre of mass relative to the inertial coordinate system expressed in the inertial frame (SP) followed by the angular acceleration of the payload body expressed in the payload coordinate system (PA)[7].

thlout double precision variable containing the highline cable tension.

tihout double precision variable containing the in-haul cable tension.

tohout double precision variable containing the out-haul cable tension.

hlpntsoutX(20) double precision vector containing the x coordinates of the nhlpntsin interpolated highline cable points relative to the origin of the inertial coordinate system and expressed in the inertial coordinate frame (SP).

hlpntsoutY(20) double precision vector containing the y coordinates of the nhlpntsin interpolated highline cable points relative to the origin of the inertial coordinate system and expressed in the inertial coordinate frame (SP).

hlpntsoutZ(20) double precision vector containing the z coordinates of the nhlpntsin interpolated highline cable points relative to the origin of the inertial coordinate system and expressed in the inertial coordinate frame (SP).

3.3.4 Verification

A development environment for RAS (DERAS) was developed that exercised the RASEQII module in the same way as the HLA federation. The environment was used

[6]Angular velocity data is expressed in body-fixed coordinates to provide consistency with the composite state vector (Table 5), the form of which is governed by the use of Newton-Euler equations.

[7]Angular acceleration data is expressed in body-fixed coordinates to provide consistency with the time derivative of the composite state vector (Table 5), the form of which is governed by the use of Newton-Euler equations.

extensively to verify the RAS equipment model and to ensure that its various features and branches work correctly under a wide range of input conditions including no payload mass, traversing from the receiving ship back to the supply ship, initializing the payload orientation relative to different ships or the inertial frame, etc. All indications suggest that the RASEQII module achieves its intended purpose and correctly implements the mathematical model of the RAS equipment described in this report.

3.4 Input File

3.4.1 Contents

The upper and lower portions of a sample input file are shown in Figures 23 and 24 respectively. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide detailed description of the meaning of the specific contents of the file.

```
1  Coordinate systems:
2  2                              Euler angle sequence used for incoming ship motion and payload initial conditions (1=XYZ;2=ZYX)
3  1                              Euler angle sequence used for output payload orientation (1=XYZ;2=ZYX)
4  Supply ship attachment points:
5  0.     -4.75    18.0           highline attachment point
6  0.     -5.      17.5           in-haul attachment point
7  0.     -5.      18.5           out-haul attachment point
8  Receiving ship attachment points:
9  40.      4.75    15.           highline attachment point
10 40.      5.      14.75         in-haul attachment point
11 40.      5.      15.25         out-haul attachment point
12 Payload parameters:
13 1                              flag indicating whether a payload body is present (0=highline only;1=payload present)
14 0.        0.       2.8702      highline attachment point
15 0.        0.       2.6202      in-/out-haul attachment point
16 1000.                          mass
17  670.     0.       0.          mass moment of inertia matrix (Ixx,Ixy,Ixz)
18         540.       0.          mass moment of inertia matrix (    Ixy,Ixz)
19                    860.         mass moment of inertia matrix (        Ixz)
20 250.  250.   250.               payload translational viscous damping coefficients
21 025.  025.   025.               payload rotational viscous damping coefficients
22 Ram tensioner parameters:
23 12.    10.    24.               wire rope run lengths (Lad,Ldg,Lde0)
24 1.379d7  0.180856  6           ram nominal pressure, cylinder diameter, number of cable falls
25 0.05  1.0                      highline tension friction factor, cable velocity threshold beyond which full frictional tension is reached
26 1.693557                       nominal system air volume
27 1.5   0.1                      ram deadband length Delta_db, Delta_tol
28 0     1.d6                     flag enabling cable stretch (0=off,1=on),AE of the highline cable
29 1.9   2.0   1200000.           extension bump stop contact distance, ram extension limit, bump spring stiffness
30 8.333 0.0                      winch speed gain cwinch (m/s/m), winch speed integral gain ki
31 3.81  71168.  93000.           maximum highline winch cable speed, Thlmax, available highline winch power
32 In-/out-haul system parameters:
33 20.                            time at which commanded traverse velocity should be released from zero
34 4                              number of command points (maximum=20)
35 0.0   0.2   0.8   1.0          payload position ratio from the supply ship to the receiving ship
36 0.8   0.8   0.8   0.8          ratio of maximum traverse speed commanded
37  4450.      6.1                nominal in-/out-haul cable pretension, in-/out-haul cable design speed
38 50000.  50000.                 winch controller gains between tension and velocity error (in-haul,out-haul)
39 26700.  26700.                 maximum cable tension before slip (in-haul,out-haul)
40     6.1     6.1                maximum winch cable take-up rates (in-haul,out-haul)
41 93000.  93000.                 maximum winch power (in-haul,out-haul)
```

Figure 23: *Sample RASEQII input file (upper portion of file).*

```
42 Integration parameters:
43 1       20.                  output flag (0=all outputs,1=zero kinetic output;2=zero kinematic and kinetic output), tStartForces
44 9.81                         acceleration due to gravity (consistent with Imperial or metric units)
45 0.01    10                   time step,number of internal time steps per output time step
46 Initial conditions:
47 0.                           initial ram compression
48 0.                           accumulated integral ram stroke
49 2                            payload IC coordinate system (1=space/inertial coordinates, 2=supply ship, 3=receiving ship)
50 0.       -5.1      18.       relative position vector to highline/payload interface point
51 0.        0.        0.       Euler angles of payload coordinate system relative to reference coordinate system
52 0.        0.        0.       relative velocity vector to highline/payload interface point from reference coordinate system origin
53 0.        0.        0.       angular velocity of payload expressed in the reference coordinate system
```

Figure 24: *Sample RASEQII input file (lower portion of file).*

Input data is categorized in eight blocks in the RASEQII.INP input file. These are:

1. coordinate system specification (Table 6);
2. supply ship attachment points (Table 6);
3. receiving ship attachment points (Table 6);
4. payload parameters (Table 7);
5. ram tensioner parameters (Table 8);
6. in-/out-haul system parameters (Table 9);
7. integration parameters (Table 10); and
8. initial conditions (Table 10).

The content of each of these data blocks is described in Tables 6 through 10. The specific tables where the data are identified are indicated in the above list.

3.4.2 Sample Parameters

Typical parameter values provided in the sample input file (Figures 23 and 24) are based to the extent possible on the existing system implementation on the PROTECTEUR as the parameters for this system are well-defined and firm. Generally, input parameters correspond to an operating pressure of 2000 psi with a 7 inch piston bore. All system parameters are controlled through the RASEQII.INP input file and can easily be changed by the user.

Table 6: RASEQII.INP *file parameters relating to Lines 1 through 11.*

Line	Input parameters
1	Text indicating the beginning of the coordinate system input data.
2	Integer flag specifying Euler rotation sequence for ship motion and payload initial conditions data (1 indicates XYZ rotation convention, 2 indicates ZYX rotation convention).
3	Integer flag specifying Euler rotation sequence for output payload orientation (1 indicates XYZ rotation convention, 2 indicates ZYX rotation convention).
4	Text indicating the beginning of the supply ship attachment point data.
5	(X,Y,Z) coordinates of the highline cable attachment point to the supply ship (Point A) expressed in the supply ship coordinate system (SU).
6	(X,Y,Z) coordinates of the in-haul cable attachment point to the supply ship (Point C) expressed in the supply ship coordinate system (SU).
7	(X,Y,Z) coordinates of the highline cable attachment point to the supply ship (Point B) expressed in the supply ship coordinate system (SU).
8	Text indicating the beginning of the receiving ship attachment point data.
9	(X,Y,Z) coordinates of the highline cable attachment point to the receiving ship (Point H) expressed in the receiving ship coordinate system (RE).
10	(X,Y,Z) coordinates of the lower out-haul cable attachment point to the receiving ship (Point J) expressed in the receiving ship coordinate system (RE).
11	(X,Y,Z) coordinates of the upper out-haul cable attachment point to the receiving ship (Point I) expressed in the receiving ship coordinate system (RE).

Table 7: RASEQII.INP *file parameters relating to Lines 12 through 21.*

Line	Input parameters
12	Text indicating the beginning of the payload parameter input data.
13	Integer flag indicating whether the simulation should include payload dynamics (0 indicates that a payload is not present, 1 indicates that a payload is present).
14	(X,Y,Z) coordinates of the highline cable interface point (Point K) relative to the payload body centre of mass (Point P) expressed in the payload coordinate system (PA).
15	(X,Y,Z) coordinates of the in-haul/out-haul cable interface point (Point L) relative to the payload centre of mass (Point P) expressed in the payload coordinate system (PA).
16	mass of the payload body including its harness and traveller.
17	First row of the upper right portion of the symmetric payload mass moment of inertia matrix containing elements I_{xx}, I_{xy}, and I_{xz} respectively expressed in the payload coordinate system (PA).
18	Second row of the upper right portion of the symmetric payload mass moment of inertia matrix containing elements I_{yy} and I_{yz} respectively expressed in the payload coordinate system (PA).
19	Third row of the upper right portion of the symmetric payload mass moment of inertia matrix containing element I_{zz} expressed in the payload coordinate system (PA).
20	Effective viscous damping coefficients applied to payload translational motion in the inertial X, Y, and Z directions respectively.
21	Effective viscous damping coefficients applied to payload rotational motion in the payload frame (PA) X, Y, and Z directions respectively.

Table 8: RASEQII.INP *file parameters relating to Lines 22 through 31.*

Line	Input parameters
22	Text indicating the beginning of the ram tensioner input data.
23	Lengths of cable runs in-board of the supply ship cable attachment points for cable spans between points A and D, D and G, and D and E (with the ram in its equilibrium position) respectively.
24	Ram nominal operating pressure (P_0), ram cylinder diameter (D), and number of cable fall in the ram tensioner (NF)
25	Friction factor (γ) used for estimating ram tensioner frictional effect on tension, cable speed at which full friction is developed (V_{thresh}).
26	Nominal system total gas volume (V_0).
27	Ram deadband length within which the winch centring control is not enabled (Δ_{db}), the ram centring tolerance within which the winch centring control is disabled (Δ_{tol}).
28	Integer flag enabling the inclusion of cable stretch in the highline cable model, product of the highline cable effective cross-sectional area and modulus of elasticity ($A_c E_c$).
29	Ram extension travel at which the extension bumper spring is first contacted (L_{spring}), extension distance at which a hard stop is reached (L_{stop}), and linear bumper spring stiffness value (k_{bump}).
30	Highline winch controller proportional gain relating winch cable speed to ram centring error (c_{winch}), winch speed integral gain relating winch cable speed to the integral of ram centring error (k_i).
31	Highline winch design cable speed ($\dot{q}_{c\ max\ design}$), highline winch maximum tension above which cable is payed out ($T_{hl\ max}$), highline winch maximum power ($P_{hl\ winch\ max}$).

Table 9: RASEQII.INP *file parameters relating to Lines 32 through 41.*

Line	Input parameters
32	Text indicating the beginning of the in-/out-haul system input data.
33	Simulation time at which commanded traverse operation is allowed to proceed.
34	Integer number of control points used for specifying traverse speed profile as a function of the ratio of displacement progress from the supply ship to the receiving ship ($r_{payload}$) (maximum of 20 points).
35	The payload position ratio corresponding to each of the traverse control points.
36	The commanded fraction of maximum traverse speed corresponding to each of the traverse control points.
37	Nominal cable pretension for the in-/out-haul cables ($T_{pretension}$), in-/out-haul cable maximum design speed ($v_{tuiohmax}$).
38	In-haul winch proportional controller gain between tension and velocity error (G_{ih}), out-haul winch proportional controller gain between tension and velocity error (G_{oh}).
39	Maximum cable tension before winch slip for the in-haul and out-haul winches respectively ($T_{ih\,max}$ and $T_{ih\,max}$ respectively).
40	Maximum winch cable take up rates for the in-haul and out-haul cable winches respectively ($v_{tuihmax}$ and $v_{tuihmax}$).
41	Maximum available winch power for the in-haul and out-haul winches respectively (P_{ihmax} and P_{ohmax}).

Table 10: RASEQII.INP *file parameters relating to Lines 42 through 53.*

Line	Input parameters
42	Text indicating the beginning of the integration parameter input data.
43	Integer output flag indicating desired output from the RAS equipment model (0 indicates that all force and kinematic outputs should be returned as evaluated, 1 indicates that kinetic output should be zeroed, and 2 indicates that both kinematic and kinetic output should be zeroed), the simulation time beyond which kinetic data should be output - kinetic output will be zeroed prior to this time.
44	Acceleration due to gravity specified in units consistent with all other inputs (metric or Imperial).
45	Federation time step, number of intermediate time steps to be taken within the RAS gear module per federation time step.
46	Text indicating the beginning of the initial condition input data.
47	Initial compression of the ram from its nominal position.
48	Initial accumulated integral error in ram position.
49	Integer flag indicating the reference coordinate system in which payload initial conditions are specified (1 indicates the inertial frame (SP), 2 indicates the supply ship frame (SU), 3 indicates the receiving ship frame (RE)).
50	(X,Y,Z) components of the relative position vector from the origin of the reference coordinate system to the highline/payload interface point (Point K).
51	(X,Y,Z) Euler angles specifying the orientation of the payload body relative to the reference coordinate system (radians).
52	(X,Y,Z) components of the relative velocity vector from the origin of the reference coordinate system to the highline/payload interface point (Point K).
53	(X,Y,Z) components of the payload angular velocity expressed in the reference coordinate system.

4 Testing

As part of the project a test plan was developed, which provided a high-level overview of the testing strategy and which communicated project-wide quality standards and procedures. A copy of the test plan document may be found in Annex C. The testing itself was to identify and report as many problems with the software as possible, and through this to improve its quality, and ultimately to satisfactorily verify it against the requirements identified during the course of the project. The testing concentrated on the behavior of the overall system with principal focus on the behavior of the ship and RAS gear federates. Although exhaustive testing was not possible, eleven test cases were identified covering a reasonably broad range of test scenarios. Detailed descriptions of the test case input parameters may be found in Annex C. For information about coordinate systems and units used to describe the motions of the ships and the seaway, please see Reference [7].

The first three test cases simulated a typical payload transfer from a supply ship to a receiving ship with both ships travelling in calm water at steady speeds, constant RPMs, headings of zero degrees and zero rudder angles, and with the RAS gear mounted at midships, the bow, and the stern, respectively. In the next two cases, the RAS gear was mounted at the midships, with ship headings set to zero degrees, with waves from 315 degrees and the ships were travelling in waters at sea states 2 and 4 respectively. The sixth scenario simulated the ships travelling in calm water at steady speeds, constant RPMs, heading of zero degrees and zero rudder angles, the RAS gear mounted at the midships, and the ships coupled by the RAS highline cable, but without a payload being transferred. In the seventh and eighth test cases both ships travelled with heading set to 0, in waves from 0 degrees and in sea state 5, but in regular and random waves, respectively. In the next two test cases both ships travelled in waves from 330 degrees, in sea state 5, but in regular and random waves, respectively. The last test case, Test Case 11, provides a comparison of the results generated by the ship motions code when operated either inside or outside the federation. The operational conditions specified in this final test case were based on those used in Test Case 10, the only difference being that the RAS gear did not provide a mechanical coupling between the two vessels.

For each of the first 10 test cases plots are provided which show the following:

1. the change in distance between the ships (measured from the respective ship centres of gravity positions);
2. the ship trajectories;
3. the ship speeds; and
4. the ship headings.

The results presented for Test Case 11 provide a comparison between the supply ship motions (surge, sway, heave, roll, pitch and yaw) produced by the ship motions code when operated either inside the federation or as a standalone code (outside the federation).

4.1 Test Case 1

The first test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in calm water, both with autopilot heading of 0 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 20 knots and the propellers on the HMCS Halifax were set to turn at 184.45 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 364.273 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid-ships. On the HMCS Protecteur it was at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 5 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

It was expected that the lateral distance between the ships measured at mid-ships would decrease. The simulation results confirmed this hypothesis. During the 80 seconds of the simulation the distance between the ships decreased by 1.878 m.

Selected results are provided in Figures 25 through 28.

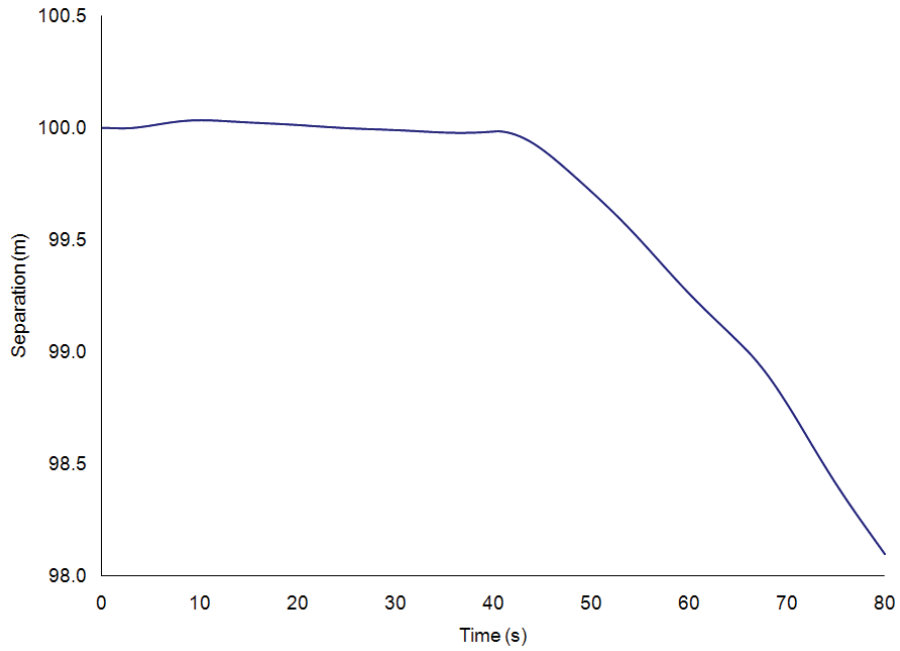

Figure 25: *Change in distance between ships during the first 80 s – Test Case 1.*

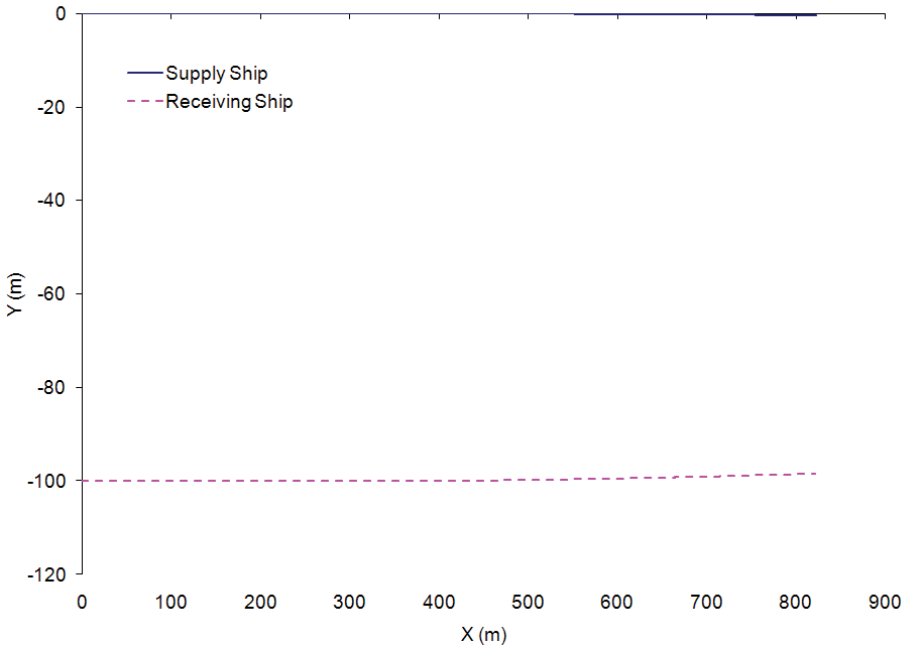

Figure 26: *Ship trajectories during the first 80 s – Test Case 1.*

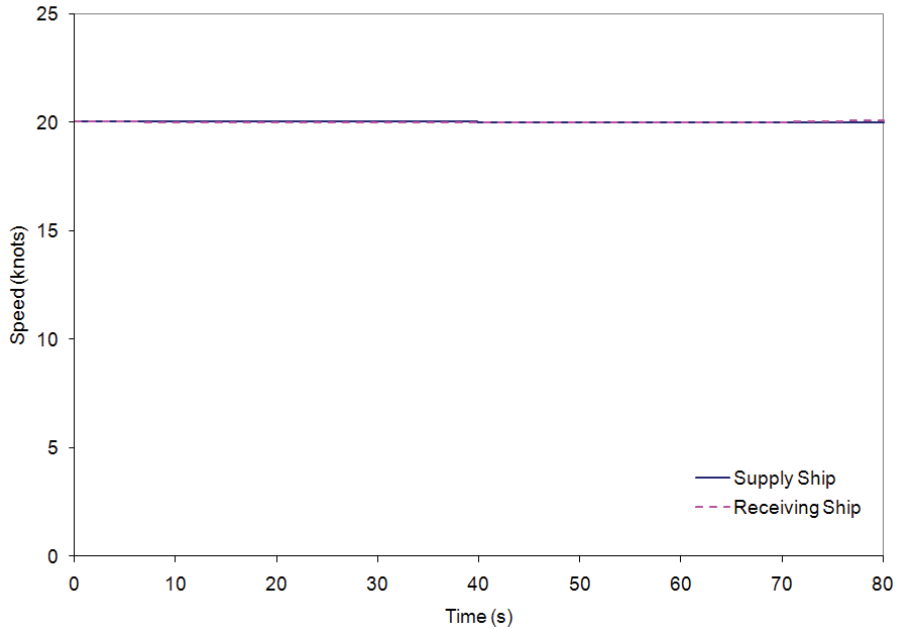

Figure 27: *Ship speeds during the first 80 s – Test Case 1.*

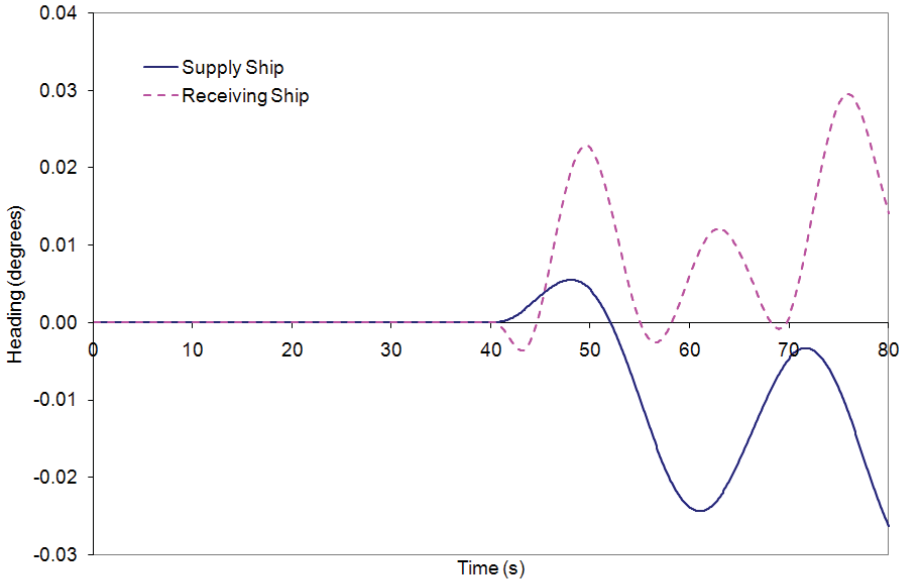

Figure 28: *Ship headings during the first 80 s – Test Case 1.*

4.2 Test Case 2

The second test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in calm water, both with autopilot heading of 0 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 20 knots and the propellers on the HMCS Halifax were set to turn at 184.45 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 364.273 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed 50 m forward of mid-ships. On the HMCS Protecteur it was at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 5 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

It was expected that the lateral distance between the ships measured at 50 m forward of mid-ships would decrease. The simulation results confirmed this hypothesis. During the 80 seconds of the simulation the distance between the ships decreased by 4.498 m.

Selected results are provided in Figures 29 through 32.

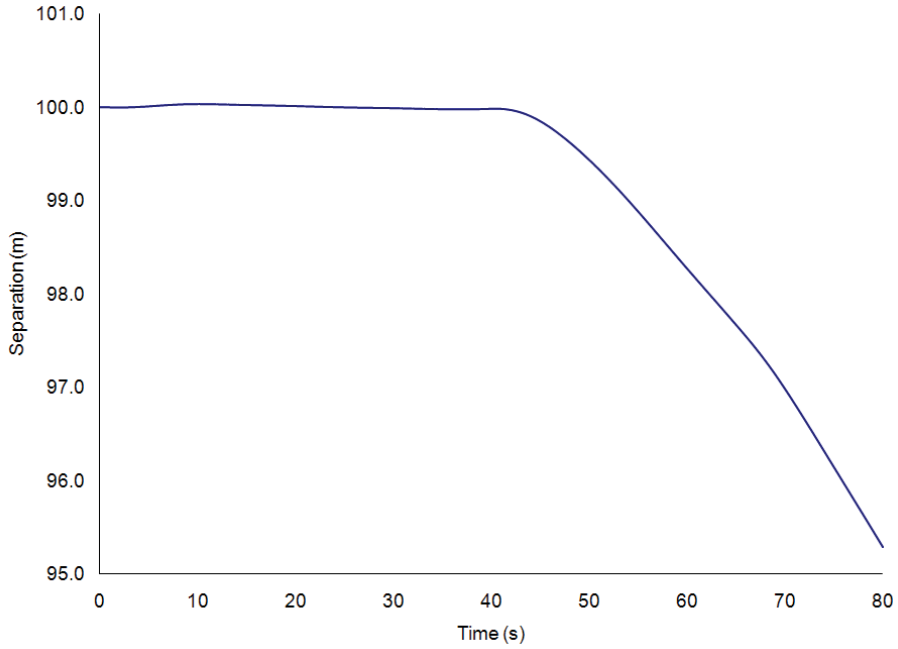

Figure 29: *Change in distance between ships during the first 80 s – Test Case 2.*

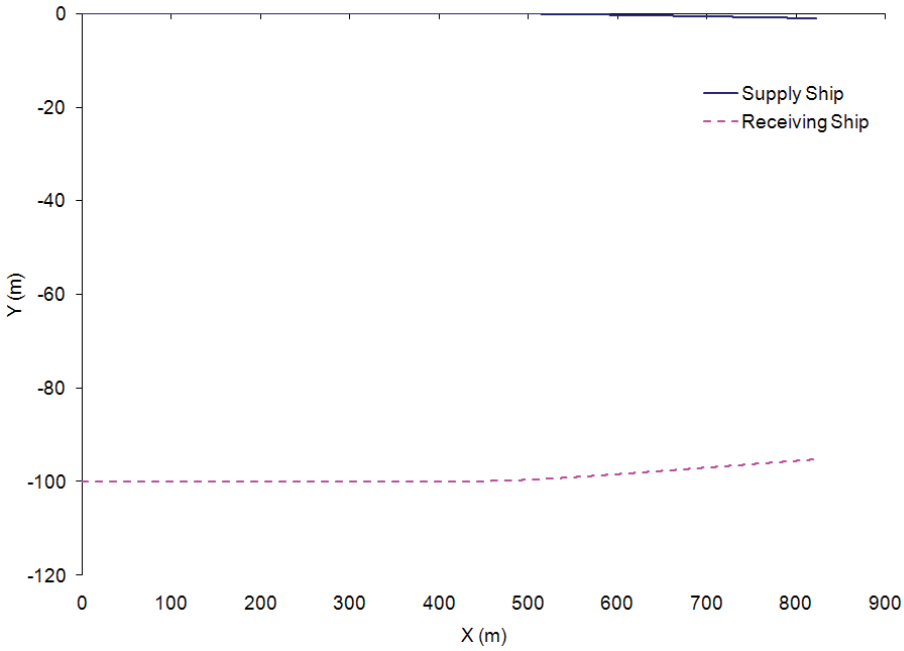

Figure 30: *Ship trajectories during the first 80 s – Test Case 2.*

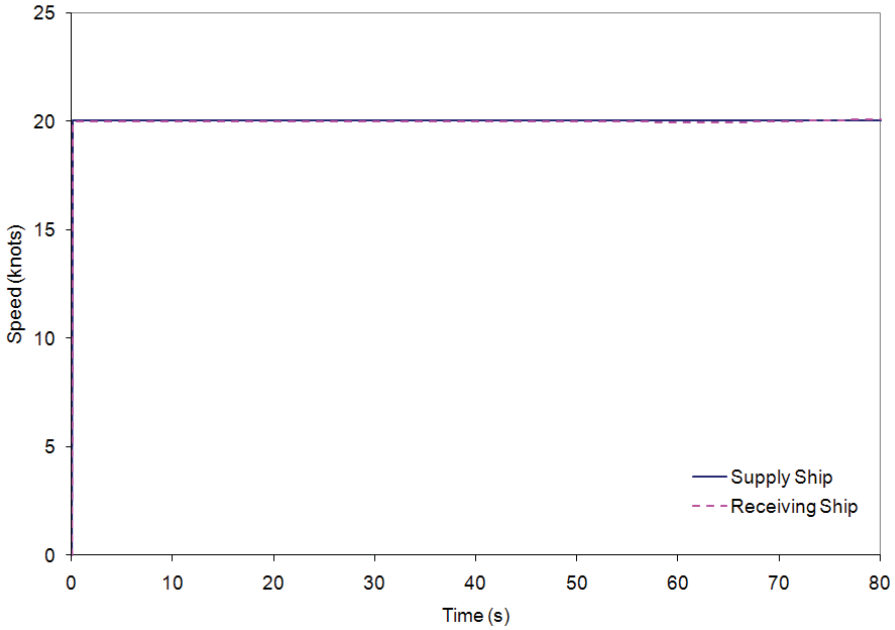

Figure 31: *Ship speeds during the first 80 s – Test Case 2.*

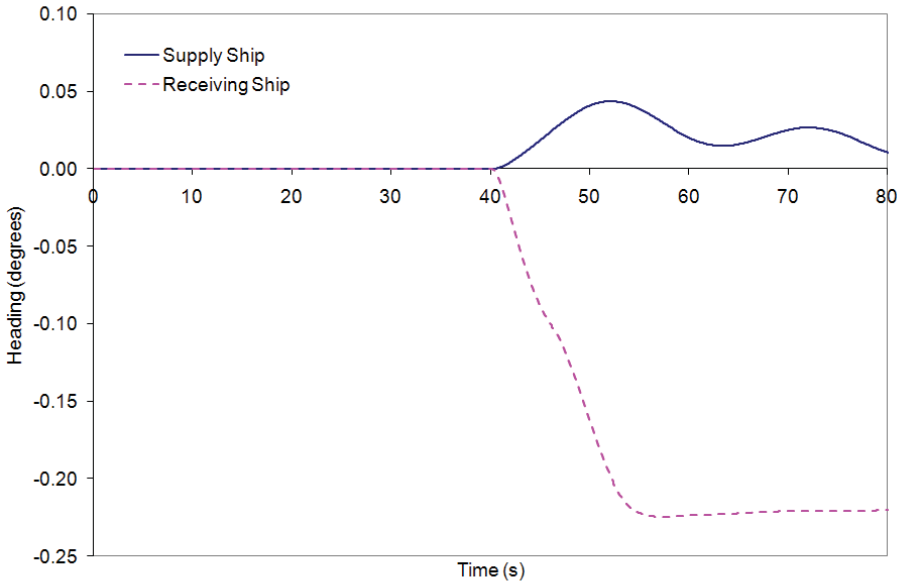

Figure 32: *Ship headings during the first 80 s – Test Case 2.*

4.3 Test Case 3

The third test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in calm water, both with autopilot headings of 0 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 20 knots and the propellers on the HMCS Halifax were set to turn at 184.45 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 364.273 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed 50 m aft of the centre of gravity. On the HMCS Protecteur it was at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 5 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

It was expected that the lateral distance between the ships would decrease. The simulation results contradicted this hypothesis. During the 80 seconds of the simulation the distance between the ships increased by 1.014m. An alternate hypothesis was proposed. The increase in distance might be explained by the action of the hydrodynamic force of the water separating the ships whose profile with respect to the heading had changed because of the forces from the RAS gear.

Selected results are provided in Figures 33 through 36.

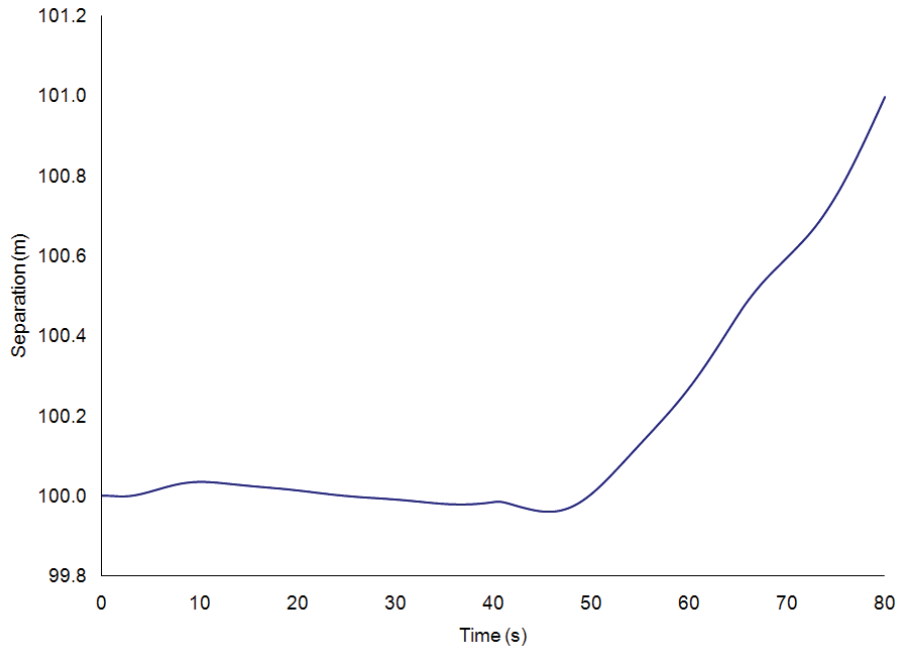

Figure 33: *Change in distance between ships during the first 80 s – Test Case 3.*

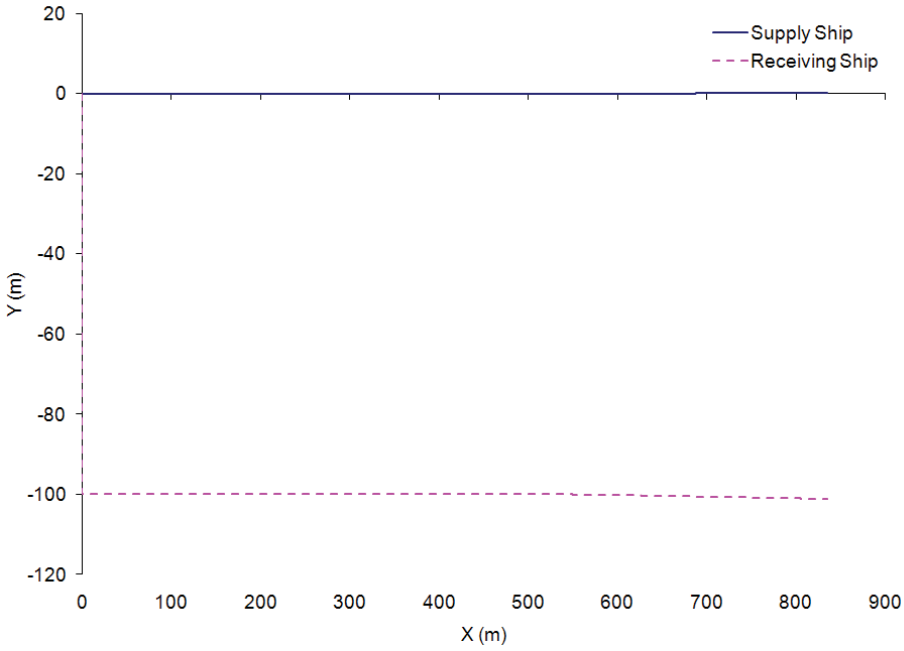

Figure 34: *Ship trajectories during the first 80 s – Test Case 3.*

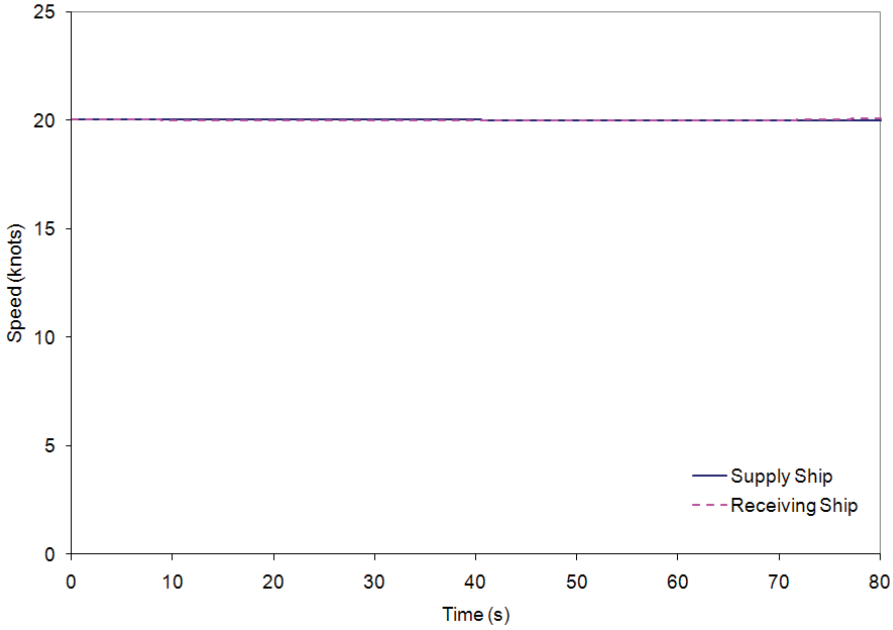

Figure 35: *Ship speeds during the first 80 s – Test Case 3.*

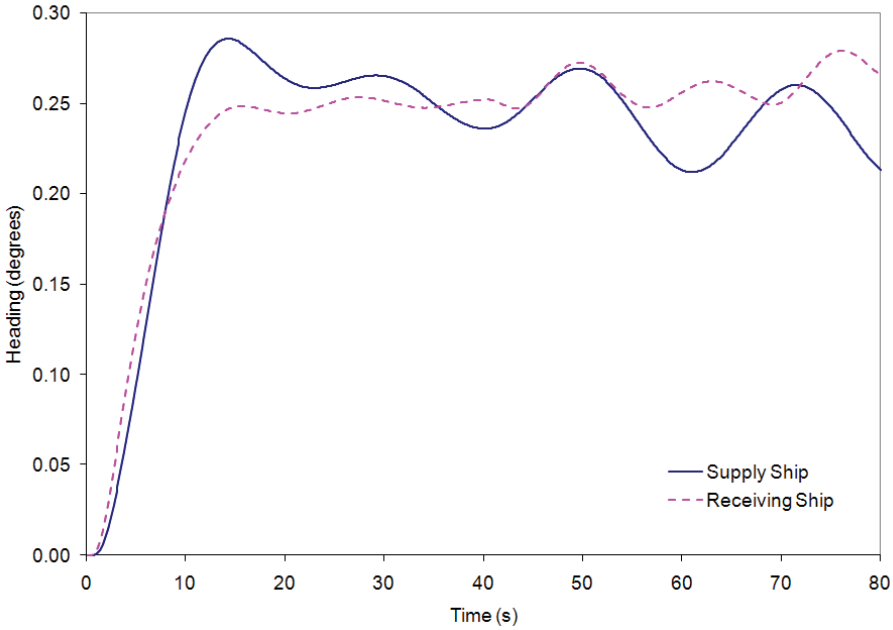

Figure 36: *Ship headings during the first 80 s – Test Case 3.*

4.4 Test Case 4

The fourth test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in water at sea state 2, both with autopilot headings at 0 degrees. The waves were set to arrive from 315 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 108.67 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 218.5 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 37 through 40.

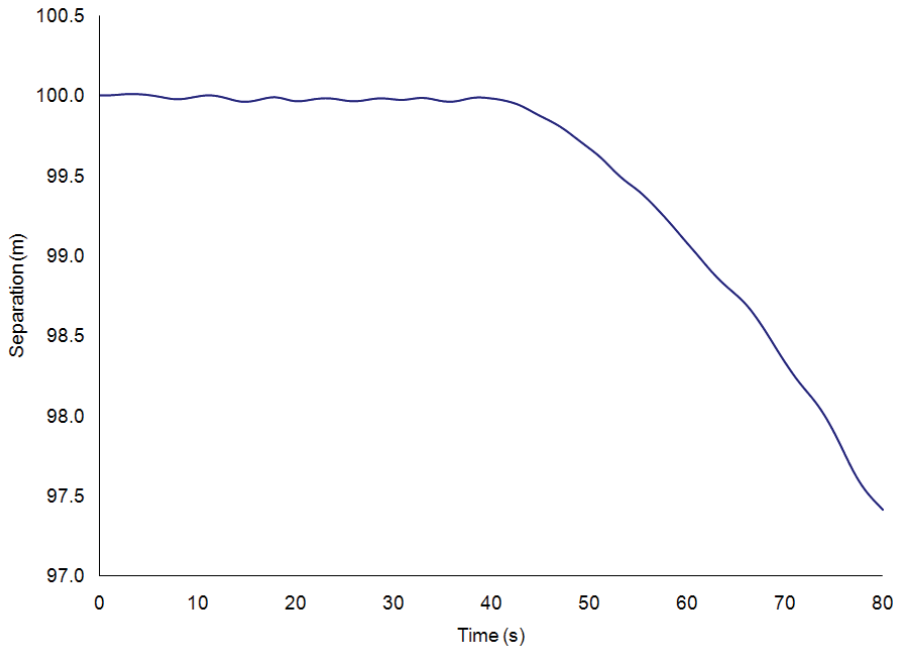

Figure 37: *Change in distance between ships during the first 80 s – Test Case 4.*

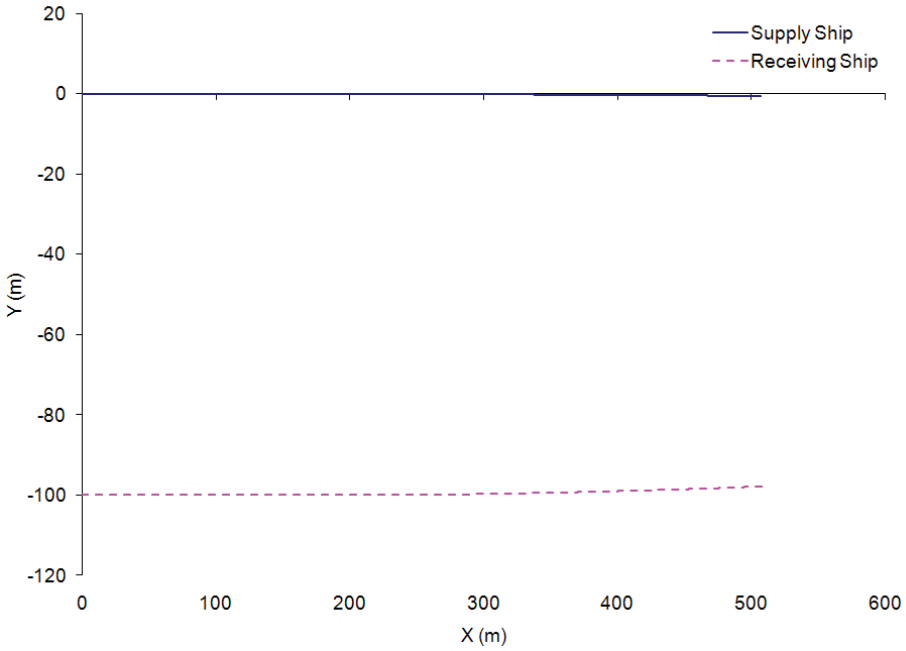

Figure 38: *Ship trajectories during the first 80 s – Test Case 4.*

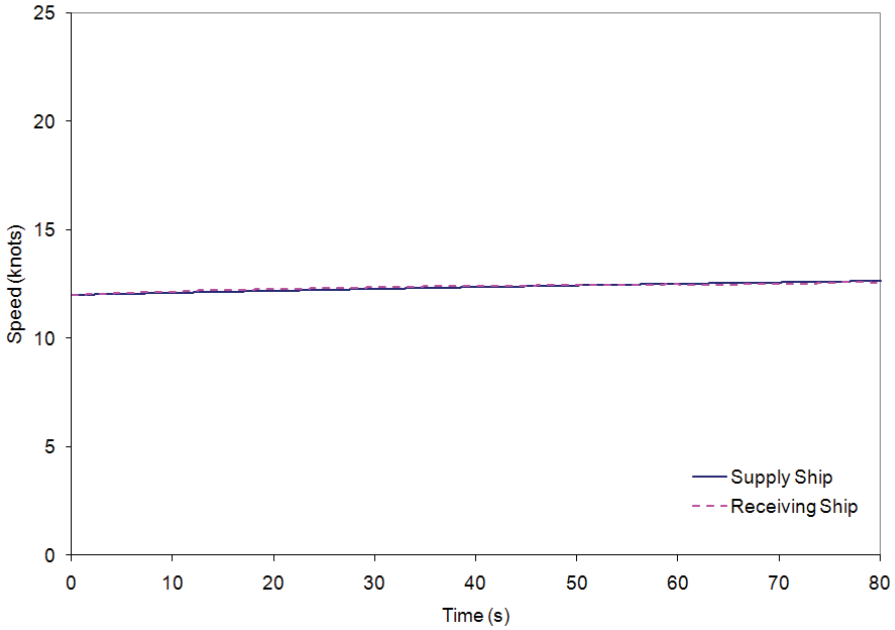

Figure 39: *Ship speeds during the first 80 s – Test Case 4.*

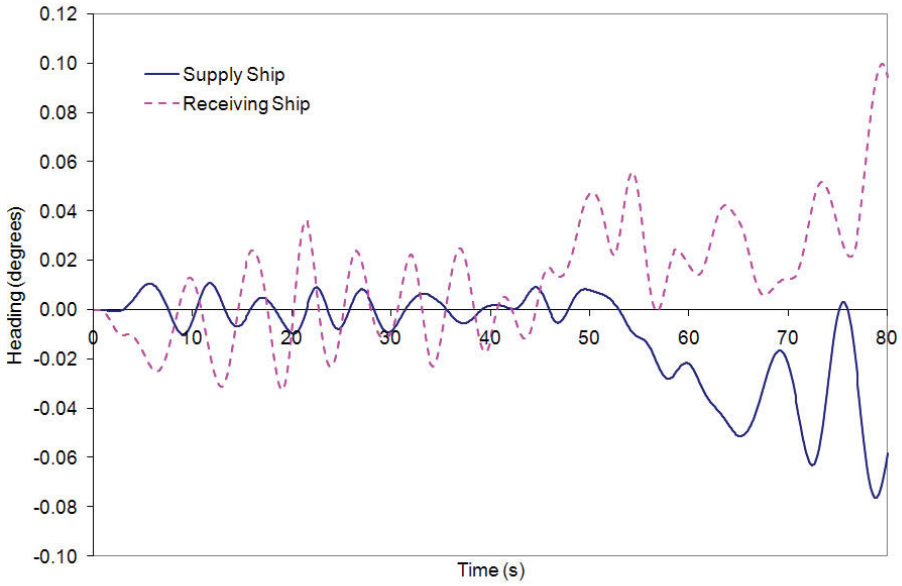

Figure 40: *Ship headings during the first 80 s – Test Case 4.*

4.5 Test Case 5

The fifth test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in water at sea state 4, both with autopilot heading of 0 degrees. The waves were set to arrive from 315 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 108.67 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 218.5 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 41 through 44.

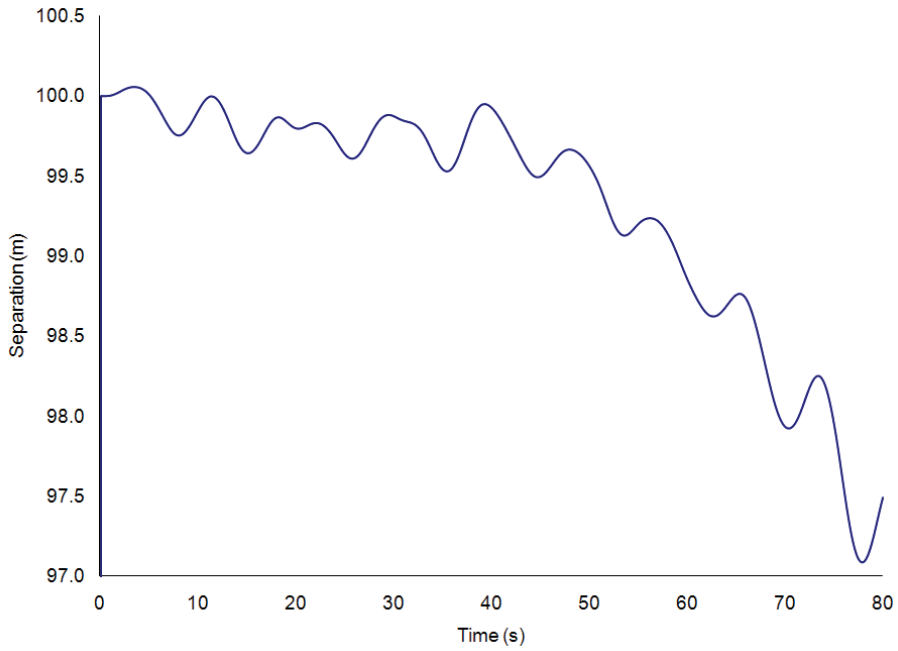

Figure 41: *Change in distance between ships during the first 80 s – Test Case 5.*

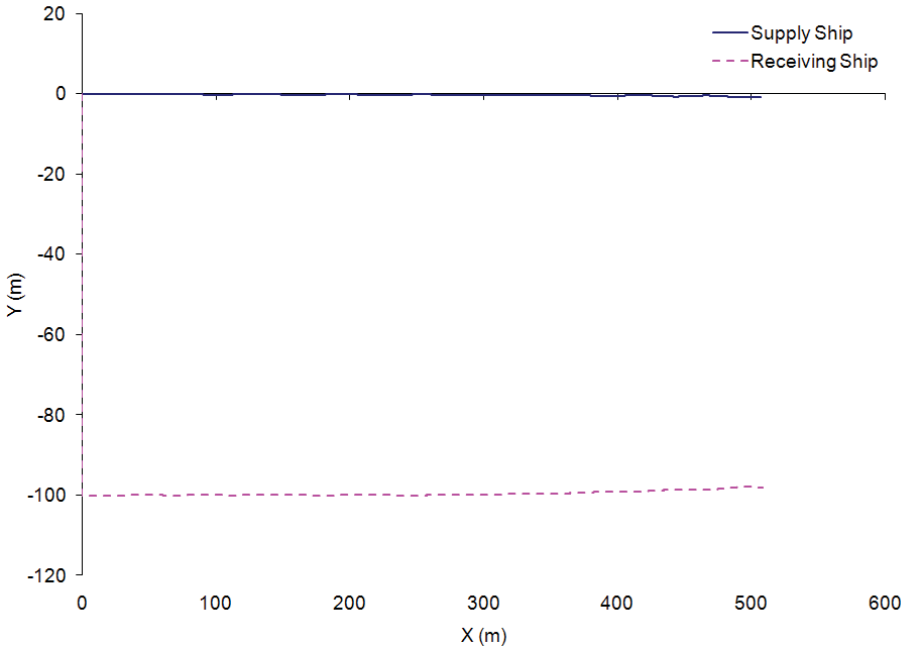

Figure 42: *Ship trajectories during the first 80 s – Test Case 5.*

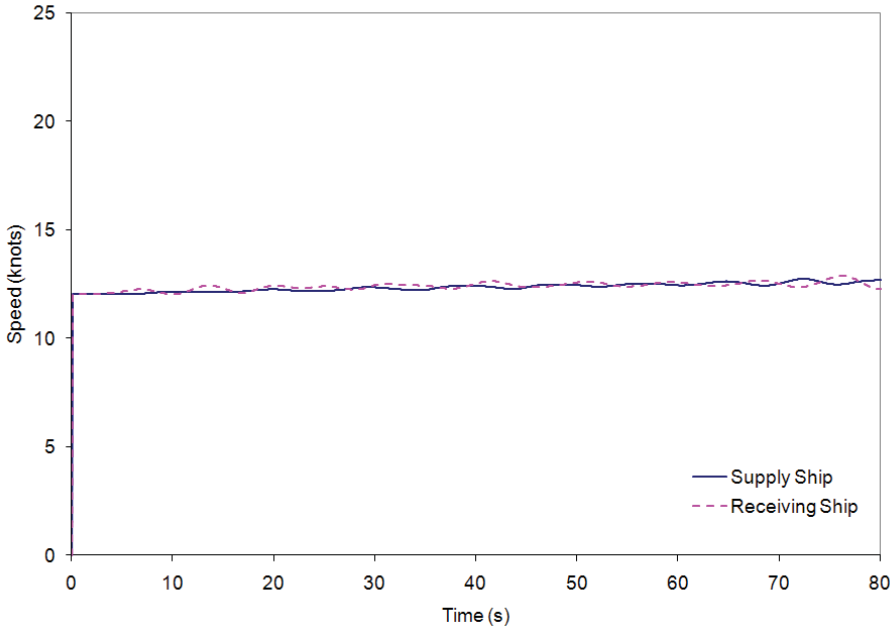

Figure 43: *Ship speeds during the first 80 s – Test Case 5.*

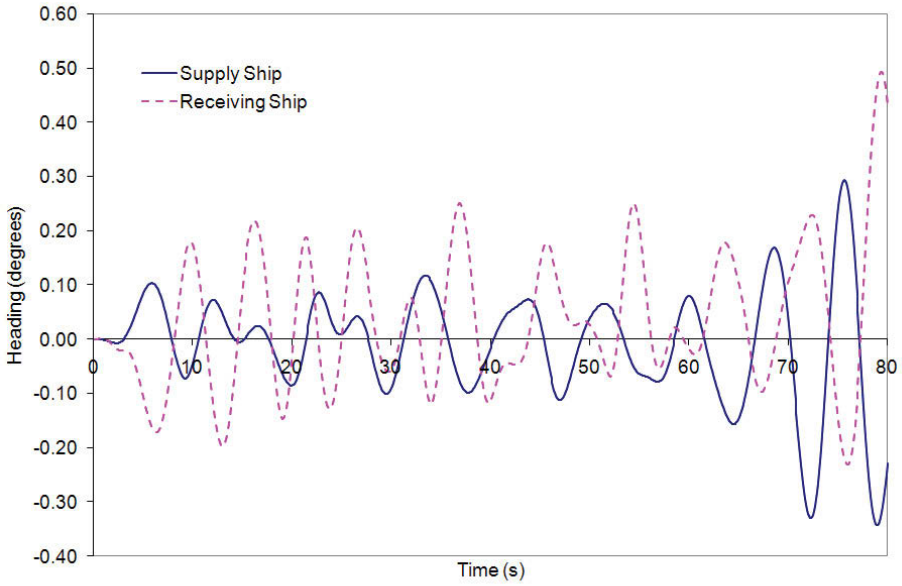

Figure 44: *Ship headings during the first 80 s – Test Case 5.*

4.6 Test Case 6

The sixth test case simulated HMCS Protecteur and HMCS Halifax coupled by the RAS highline cable, but without a payload being transferred. The ships were set to travel in calm water, both with autopilot heading of 0 degrees. The HMCS Protecteur was placed at 100 m to the port side of HMCS Halifax, measured at mid-ships and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 20 knots and the propellers on the HMCS Halifax were set to turn at 184.45 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 364.273 rotations per minute. The rudder angles for both ships' rudders was set to 0 degrees. On both ships, the RAS gear was placed at mid-ships. On the HMCS Protecteur it was at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 5 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

It was expected that the lateral distance between the ships measured at mid-ships would decrease. The simulation results confirmed this hypothesis. During the 80 seconds of the simulation the distance between the ships decreased by 1.671 m.

Selected results are provided in Figures 45 through 48.

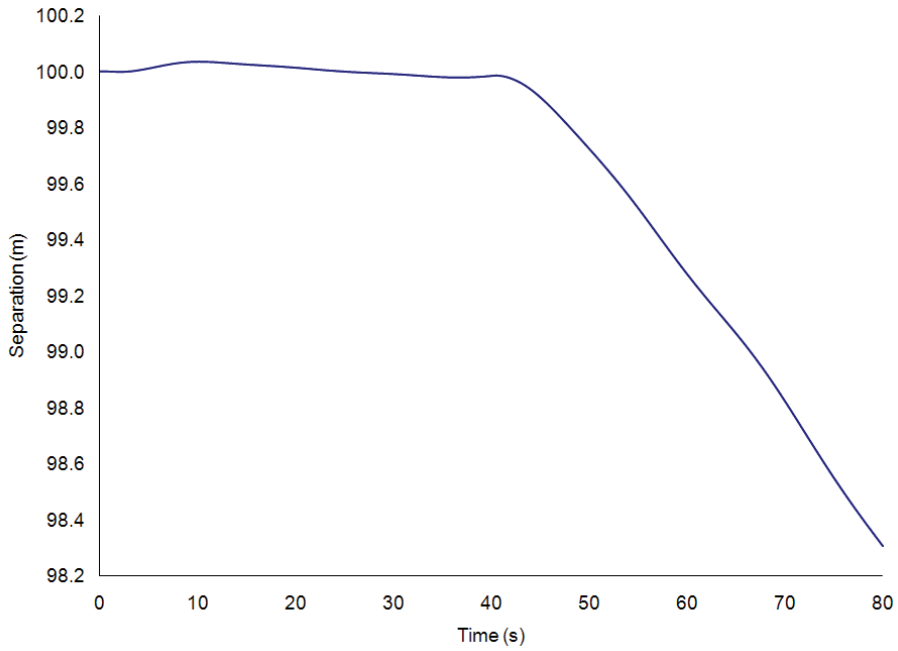

Figure 45: *Change in distance between ships during the first 80 s – Test Case 6.*

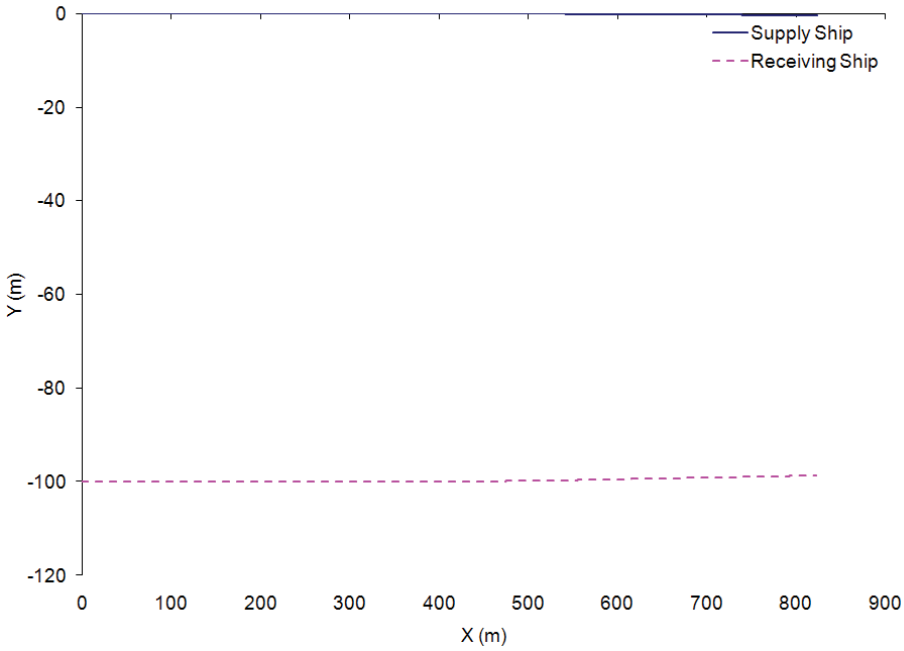

Figure 46: *Ship trajectories during the first 80 s – Test Case 6.*

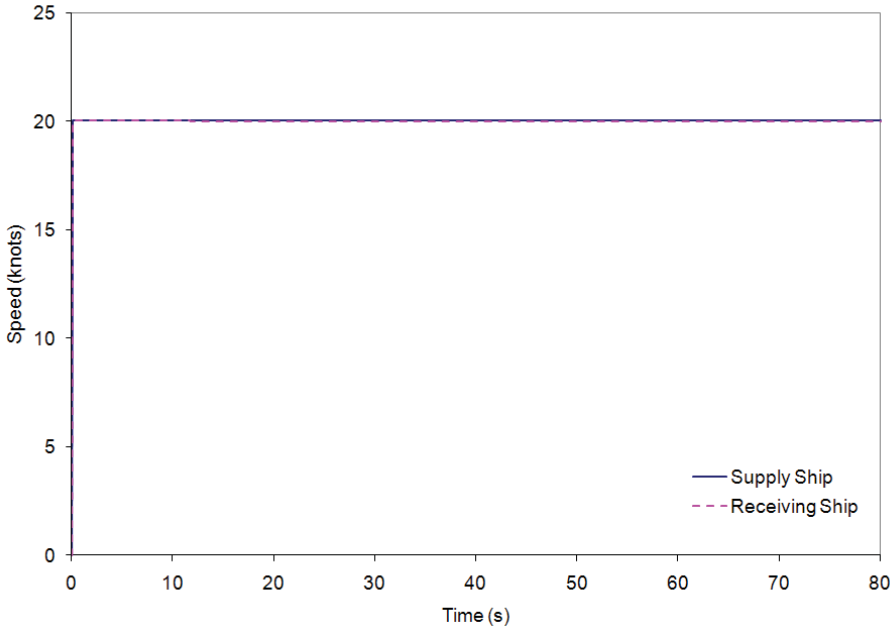

Figure 47: *Ship speeds during the first 80 s – Test Case 6.*

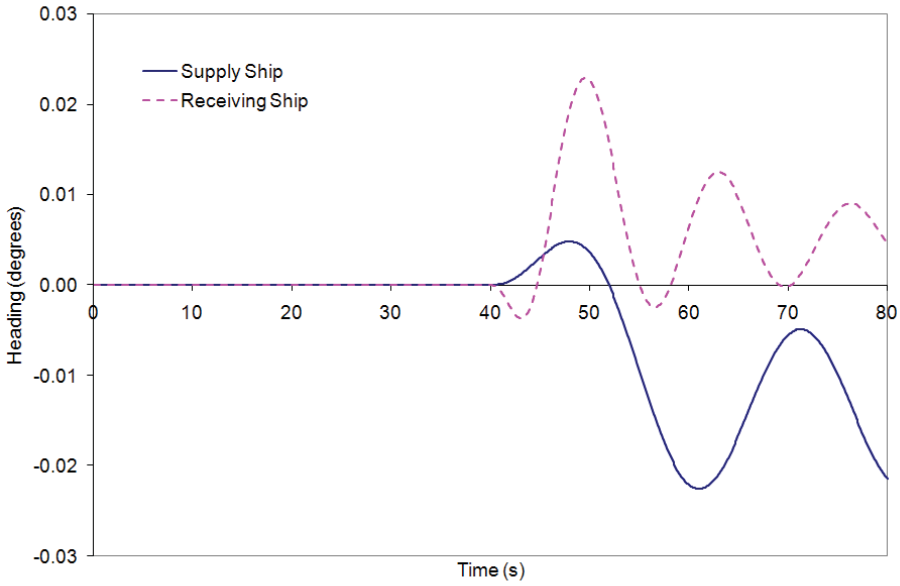

Figure 48: *Ship headings during the first 80 s – Test Case 6.*

4.7 Test Case 7

The seventh test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in sea state 5 and regular waves coming from 0 degrees, with both autopilots' headings set to 0 degrees. The HMCS Protecteur was placed at 52.25 m to the port side of HMCS Halifax, measured at mid-ships, and the ships were aligned at mid-ships and parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 111 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 219 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m port of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 49 through 52.

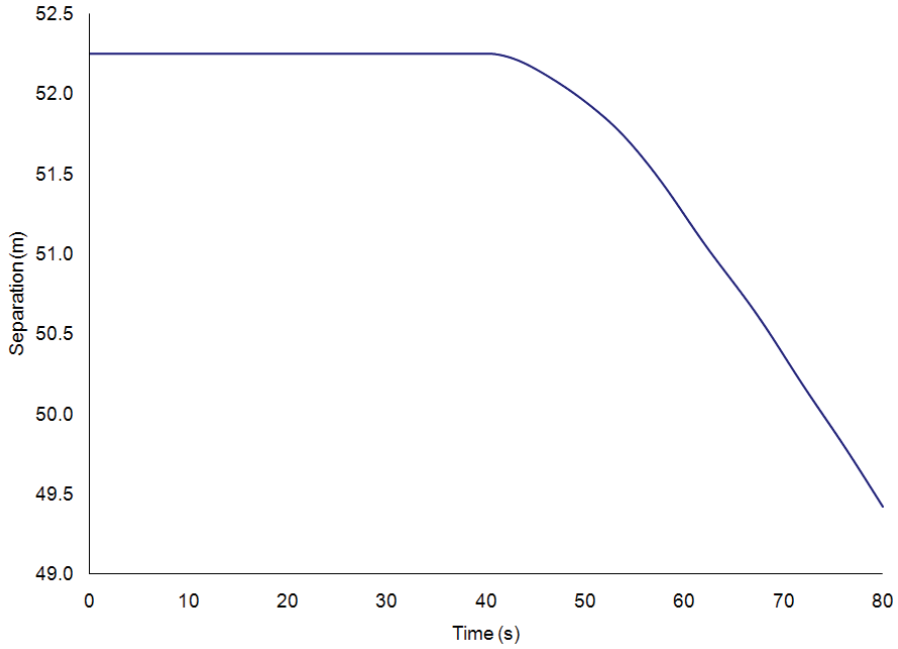

Figure 49: *Change in distance between ships during the first 80 s – Test Case 7.*

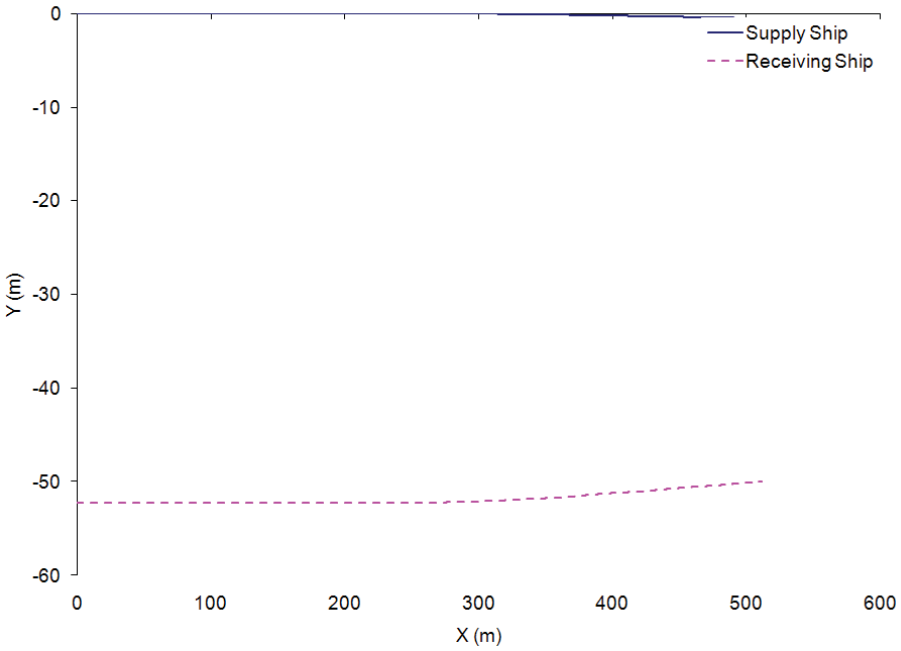

Figure 50: *Ship trajectories during the first 80 s – Test Case 7.*

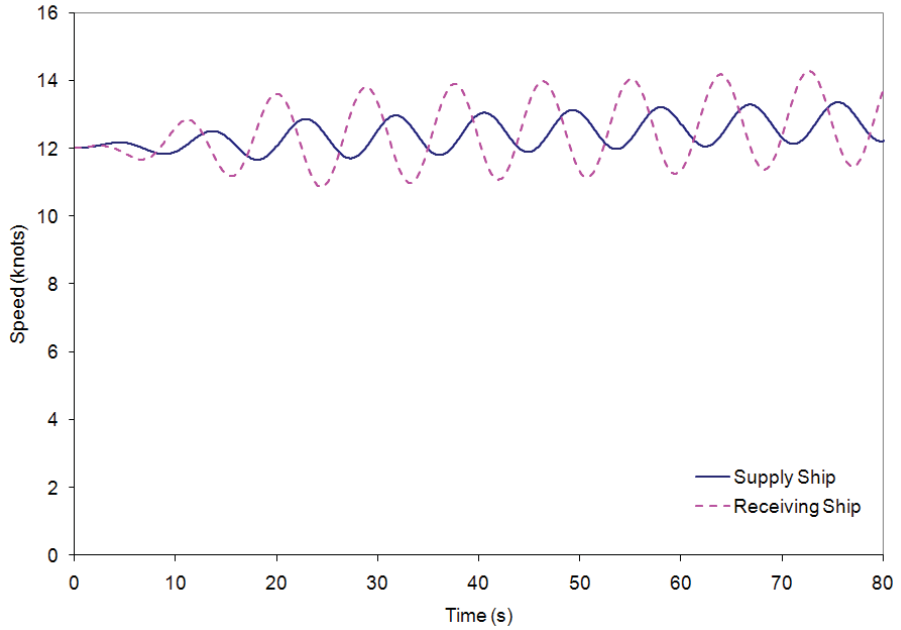

Figure 51: *Ship speeds during the first 80 s – Test Case 7.*

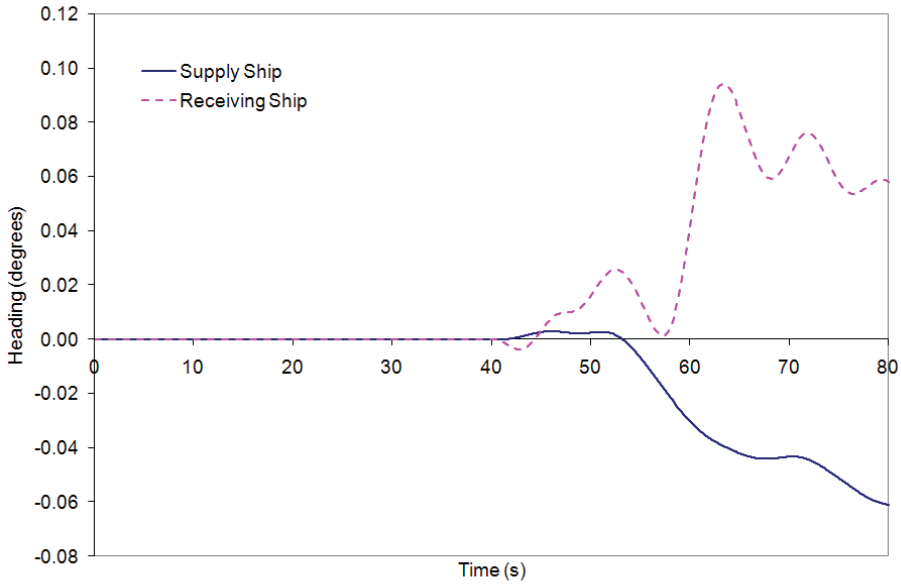

Figure 52: *Ship headings during the first 80 s – Test Case 7.*

4.8 Test Case 8

The eighth test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in sea state 5 and random waves coming from 0 degrees, with both autopilots' headings set to 0 degrees. The HMCS Halifax was placed at 52.25 m to the starboard side of HMCS Protecteur, measured at mid-ships, and the ships were aligned at mid-ships with parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 111 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 219 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid-ships. On the HMCS Protecteur it was at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above the waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 53 through 56.

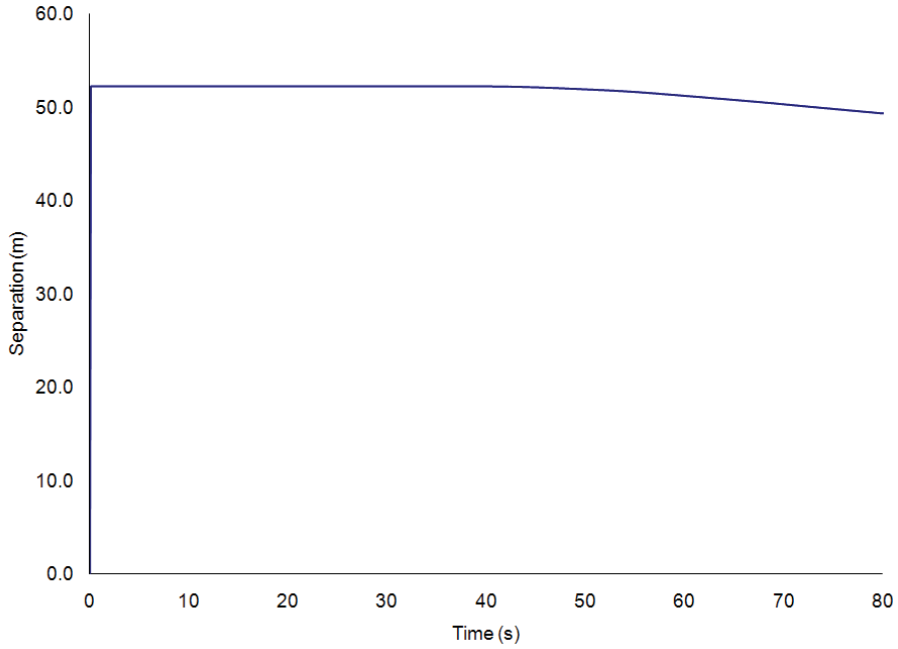

Figure 53: *Change in distance between ships during the first 80 s – Test Case 8.*

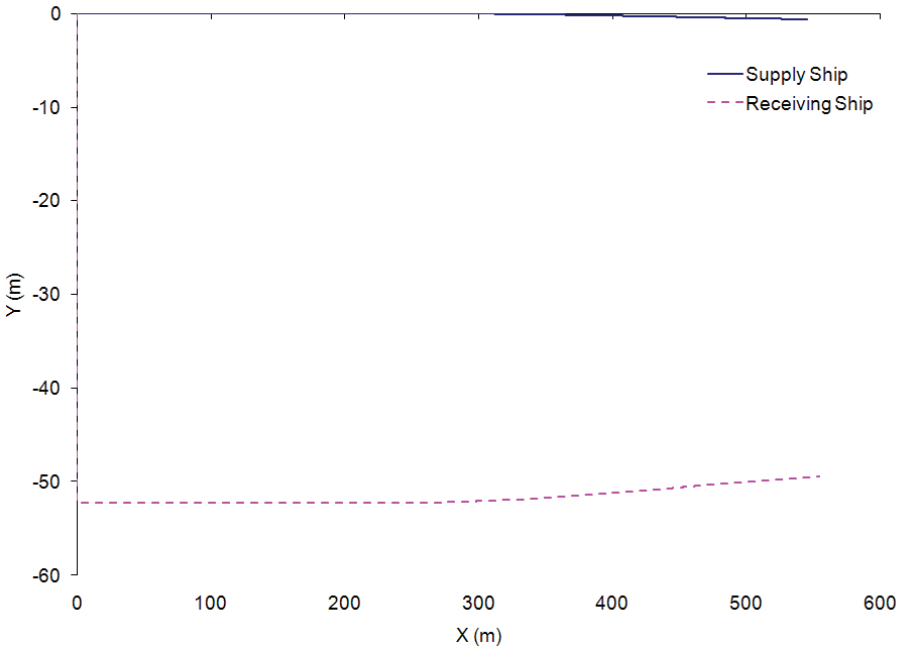

Figure 54: *Ship trajectories during the first 80 s – Test Case 8.*

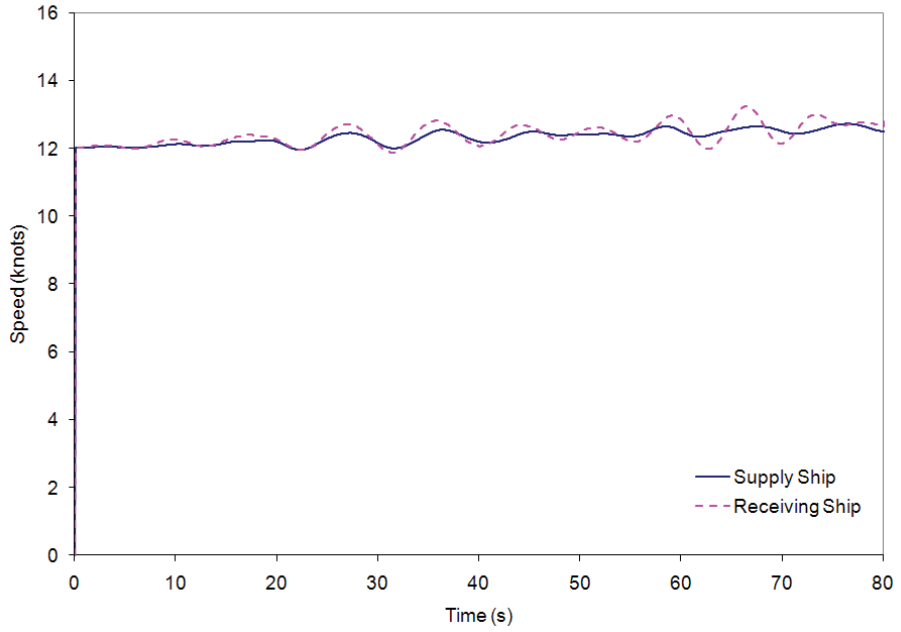

Figure 55: *Ship speeds during the first 80 s – Test Case 8.*

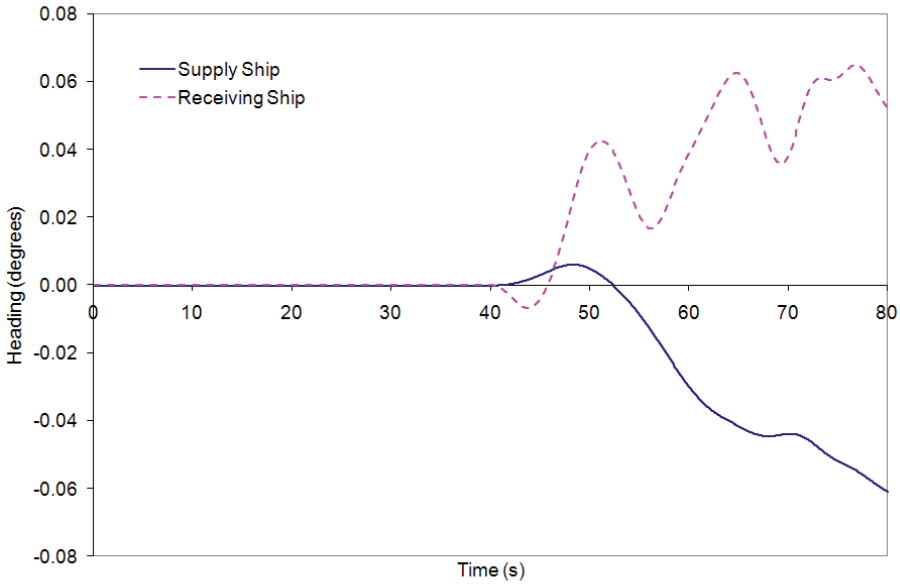

Figure 56: *Ship headings during the first 80 s – Test Case 8.*

4.9 Test Case 9

The ninth test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in sea state 5 and regular waves coming from 330 degrees, with both autopilots' headings set to 0 degrees. The HMCS Halifax was placed at 52.25 m to the starboard side of HMCS Protecteur, measured at mid-ships, and the ships were aligned at mid-ships with parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 111 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 219 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 57 through 60.

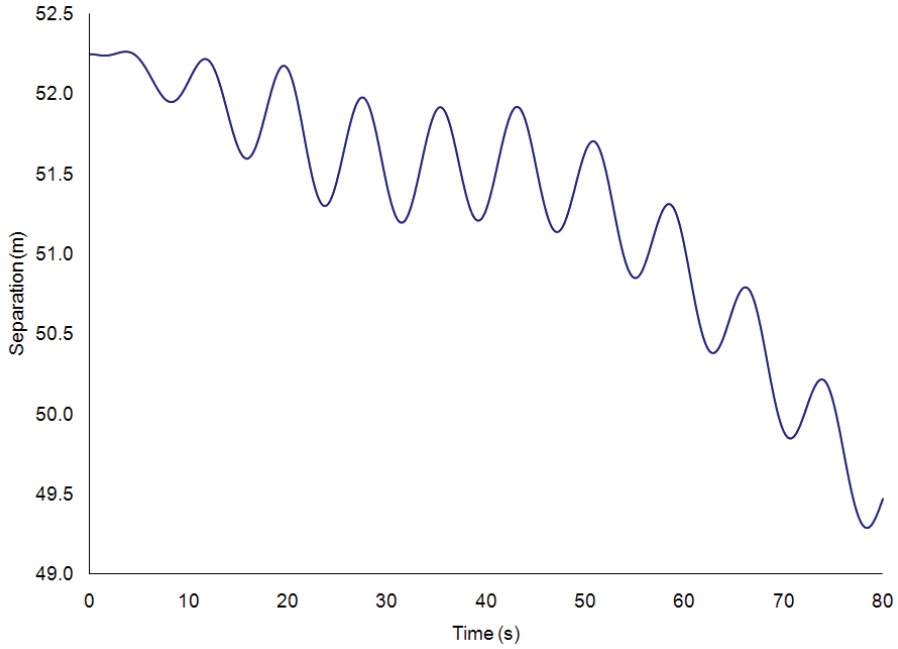

Figure 57: *Change in distance between ships during the first 80 s – Test Case 9.*

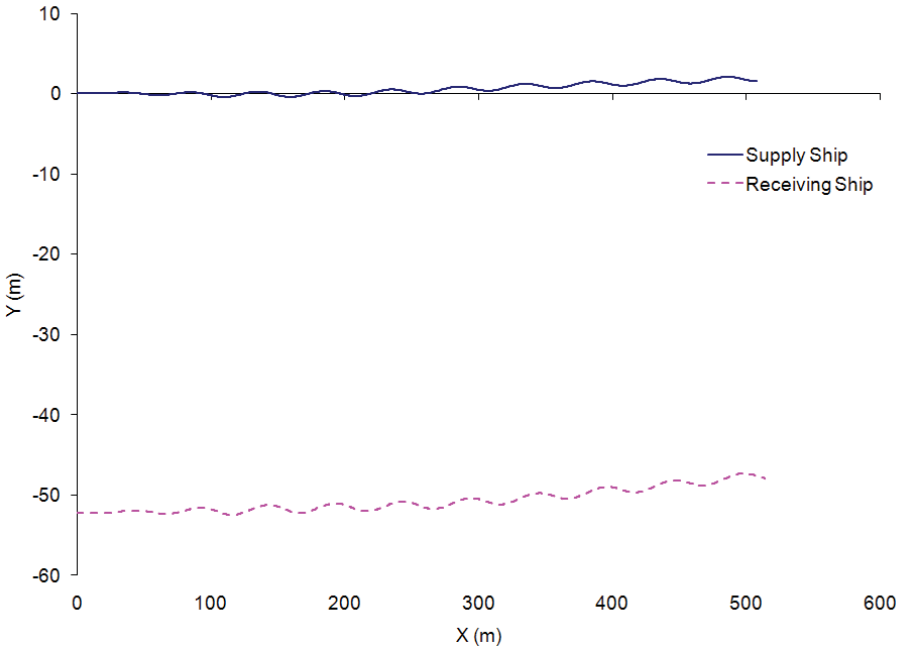

Figure 58: *Ship trajectories during the first 80 s – Test Case 9.*

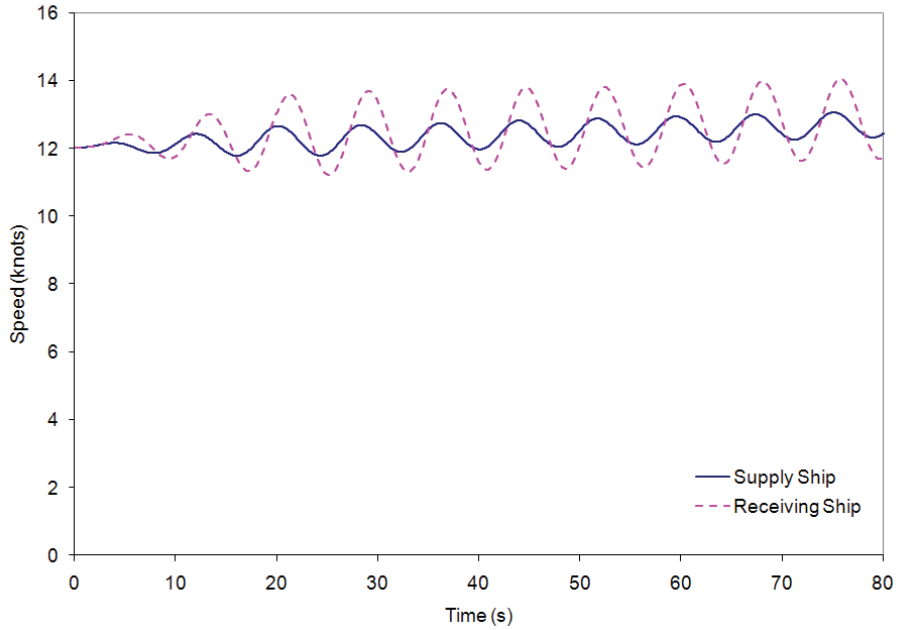

Figure 59: *Ship speeds during the first 80 s – Test Case 9.*

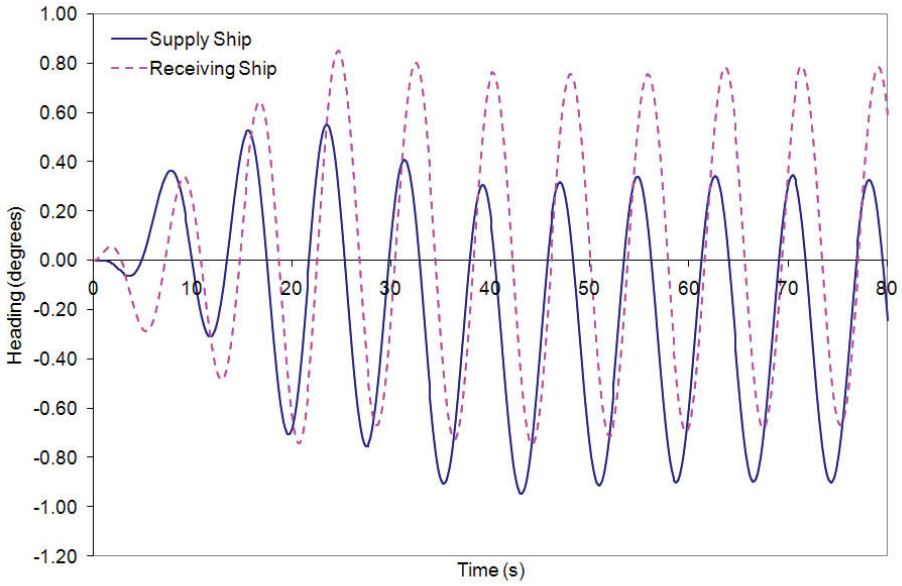

Figure 60: *Ship headings during the first 80 s – Test Case 9.*

4.10 Test Case 10

The tenth test case simulated a transfer of a 1000 kg payload from HMCS Protecteur to HMCS Halifax. The ships were set to travel in sea state 5 with random waves coming from 330 degrees, with both autopilots' headings set to 0 degrees. The HMCS Halifax was placed at 52.25 m to the starboard side of HMCS Protecteur, measured at mid-ships, and the ships were aligned at mid-ships with parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 111 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 219 rotations per minute. The rudder angles for both ships' rudders were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. The simulation was run for 80 seconds. The actual RAS operation commenced at 40 seconds.

The verification process determined that the federation of models and simulations and their associated data accurately represent the system's conceptual description and specifications. The separation measured at mid-ships appeared reasonable.

Selected results are provided in Figures 61 through 64.

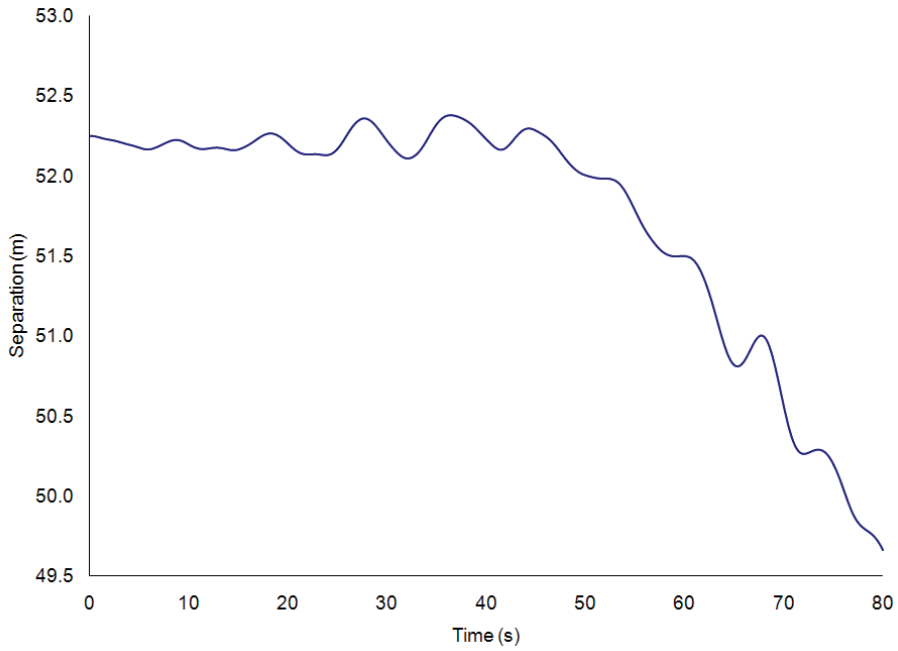

Figure 61: *Change in distance between ships during the first 80 s – Test Case 10.*

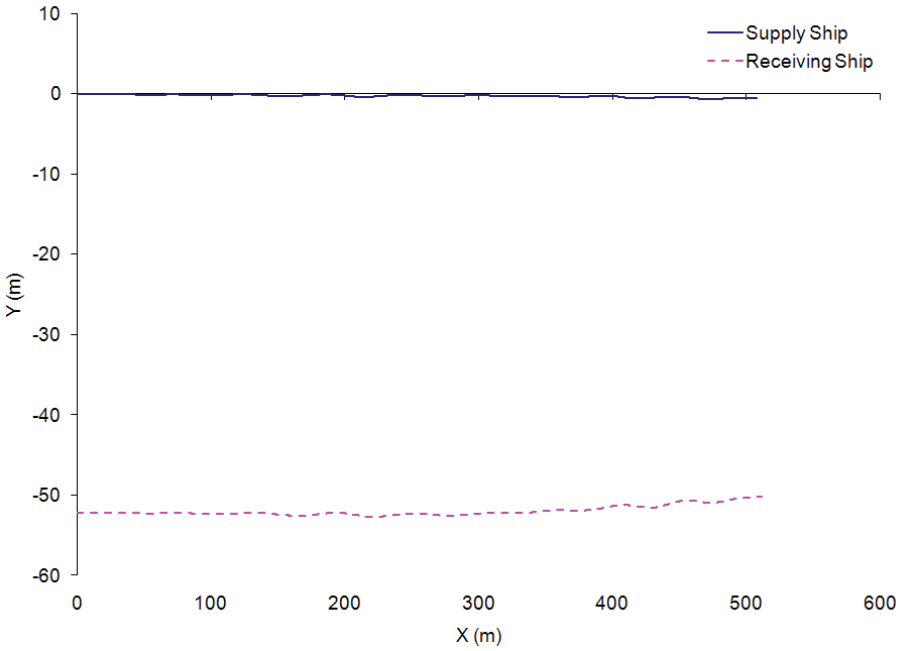

Figure 62: *Ship trajectories during the first 80 s – Test Case 10.*

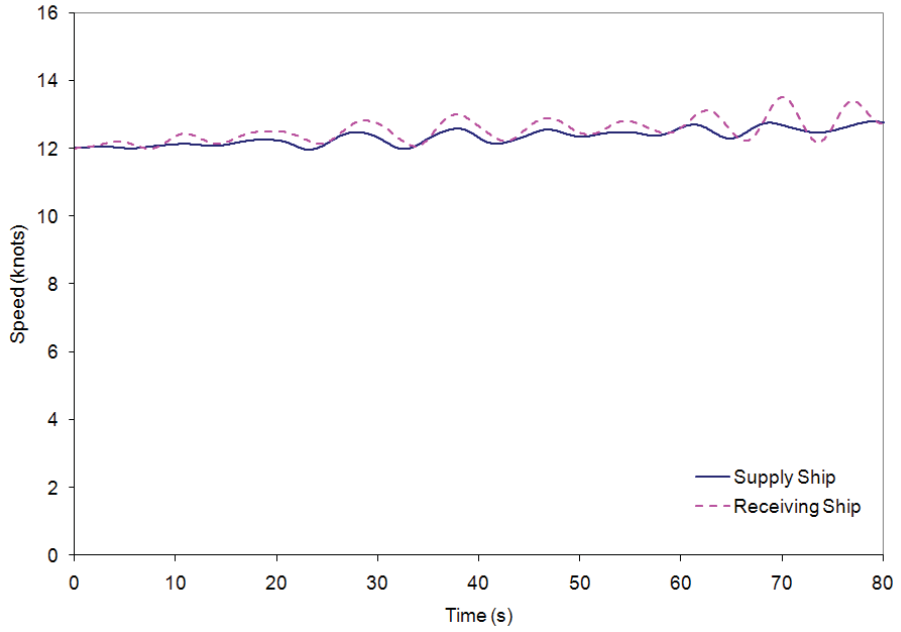

Figure 63: *Ship speeds during the first 80 s – Test Case 10.*

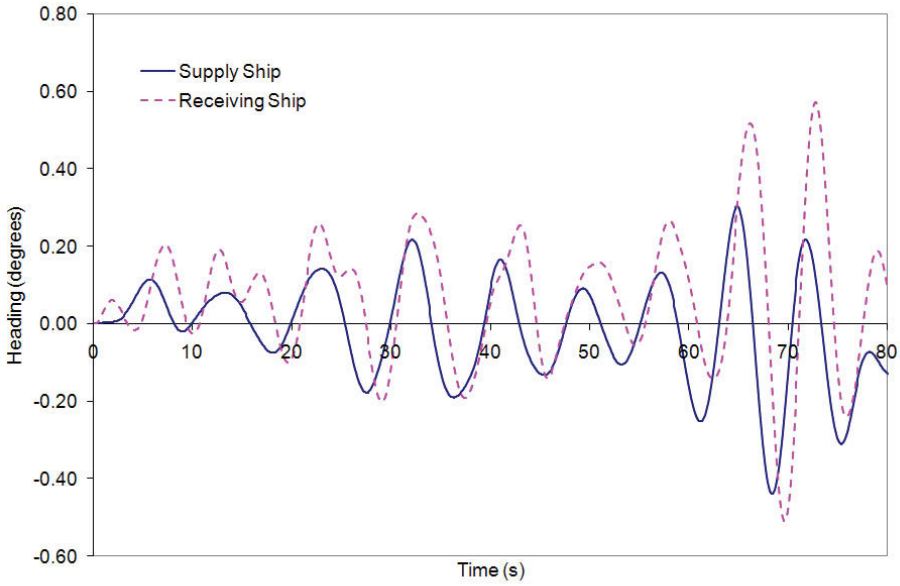

Figure 64: *Ship headings during the first 80 s – Test Case 10.*

4.11 Test Case 11

The main goal of the eleventh test was to verify, that when running the federation without the mechanical coupling provided by the RAS gear, the ship motions generated when operating ShipMo3D within the federation are identical to those generated when running the simulation outside of the federation using SM3DFreeMo. For the purposes of this study the ships were set to travel in sea state 5 with random waves coming from 330 degrees, with both autopilot headings set to 0 degrees. The HMCS Halifax was placed at 52.25 m to the starboard side of HMCS Protecteur, measured at mid-ships, and the ships were aligned at mid-ships with parallel centerlines. The speeds for both ships were set to 12 knots and the propellers on the HMCS Halifax were set to turn at 111 rotations per minute and on the HMCS Protecteur the propeller was set to turn at 219 rotations per minute. The rudder angles for both ships were set to 0 degrees. On both ships, the RAS gear was placed at mid ships. On the HMCS Protecteur it was at 4.75 m starboard of centreline and at elevation of 18 m above the waterline. On the HMCS Halifax it was at 4.75 m port of centerline and at elevation of 15 m above waterline. However, it is important to note that zero kinematic and kinetic output was published by the RAS federate (i.e. no mechanical coupling between the ships).

Figures 65 and 65 provide a comparison of the Protecteur (supply) ship motions produced by running ShipMo3D inside versus outside the federation. A review of these figures clearly shows excellent agreement between the results as all corresponding traces are superimposed, indicating that ShipMo3D has been incorporated into the federation properly.

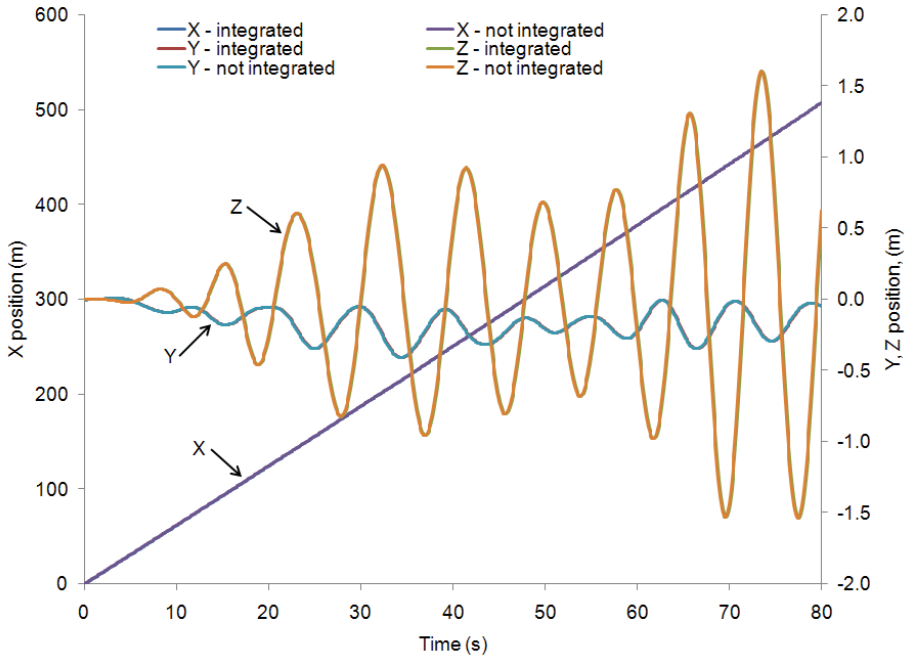

Figure 65: *Comparison of receiving ship translational motions produced inside versus outside the federation.*

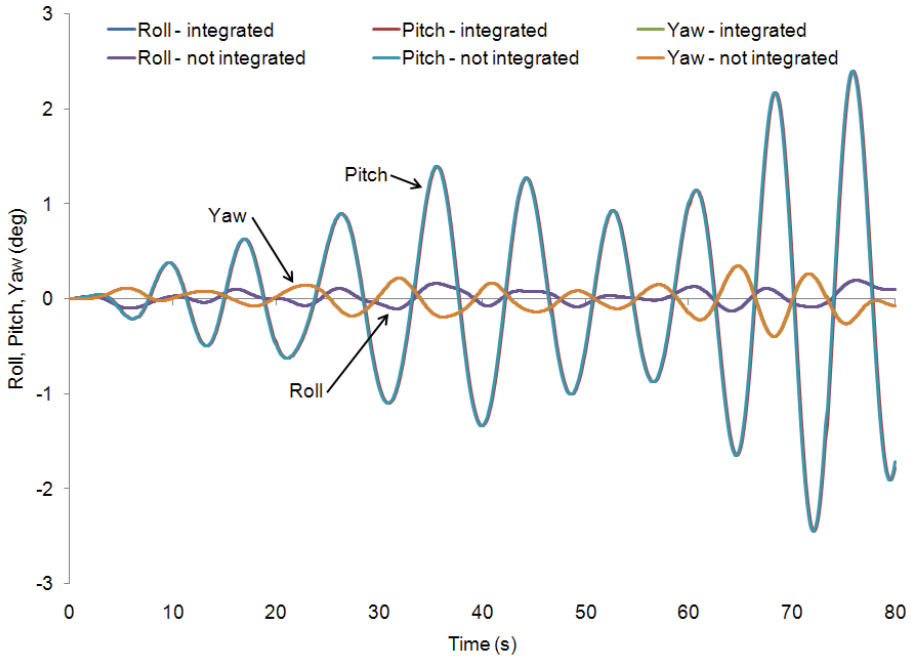

Figure 66: *Comparison of receiving ship rotational motions produced inside versus outside the federation.*

5 Conclusion

The objective of this research project was to implement a federation simulating replenishment at sea (RAS). Ultimately this simulation environment will be used to determine if adverse events occur during a RAS operation, specifically to determine if emersion of the payload and/or breakage of replenishment gear occur. In order to satisfy this goal an HLA federation containing a collection of federates representing ships, sea state, and RAS gear was developed.

Working with DRDC staff, the project team advanced the state-of-the-art in RAS simulation by developing a simulation framework that is both flexible and extensible. The physics-based modelling of the RAS equipment captures the essential elements of the dynamics and control of the dry goods replenishment process while minimizing the need for detailed design parameters that are difficult to obtain without access to detailed design documents and extensive experimentation. It is also important to note that the model is computationally efficient and versatile, with all design and operational parameters controlled through an input file.

In addition to the advances described above, a number of important conclusions can be drawn from the experiences gained during the development process. These include:

1. A time stepped (and time managed) HLA federation provides deterministic results and is fully suitable for physics-based simulations;

2. An HLA federation allows for simple modification and replacement of individual federates while minimizing the impact on other federates;

3. The latency overhead of a time-managed simulation is significant and has strong dependency on both the IP network performance and on the hardware being used;

4. The standard RPR-FOM v2.17 was designed with platform-level military training and war games in mind and provides little support for physics-based simulation of mechanical systems. A significant degree of proprietary FOM extension is required in order to adequately represent the details of such systems;

5. The RAS federation with 9 active federates behaved in a reliable way while executing over a LAN;

6. The integration of Python code with C++ is cumbersome and requires a steep learning curve for those not familiar with the Python language;

7. The integration of Fortran code with C++ is easy but imposes significant limitations on the nature of data exchange; and

8. The infrastructure created for the RAS federation can be easily leveraged for any time stepped simulation consisting of a large number of computational engines that need to interact.

In terms of future work, the project team has identified a number of tasks which would improve the fidelity of the simulation in the future. These include:

1. Comparing results from the RAS gear model against experimental data;
2. Upgrading the system level model of the RAM tensioner system cylinder to a full dynamic model that incorporates ram inertia;
3. Upgrading the highline cable model (and possibly in-haul and out-haul cables) to include the effects of cable mass such as cable sag and inertial effects; and
4. Expanding the RAS simulation to allow simulation of the hoses associated with replenishment of wet goods such as fuel and water.

References

[1] Pike, J. (1999), Underway Replenishment. Federation of American Scientists, Washington D.C. web site http://www.fas.org/man/dod-101/sys/ship/unrep.htm. Accessed November 2008.

[2] Perrault, D. (2003), Review of Architectures for Simulation of Virtual Naval Platforms, (Technical Memorandum TM 2003-193) DRDC Atlantic, Dartmouth, Nova Scotia, Canada.

[3] Bleichman, D., Langlois, R., Lichodzijewski, M., and Brennan, D. (2007), Development of a high level architecture federation of ship replenishment at sea: Phase I final report, (DRDC Atlantic Contractor Report 2007-222) DRDC Atlantic, Dartmouth, Nova Scotia, Canada.

[4] SISO (2003), Guidance, Rationale, and Interoperability Manual for Real-time Platform Reference Federation Object Model Version 2.0D17v3 (GRIM_RPR2-d17v3), SISO.

[5] Modeling, Defense and Office, Simulation, RTI 1.3-Next Generation Programmer's Guide", Version 6 ed, US Department of Defense. Was available from http://www.dmso.mil.

[6] IEEE (1998), IEEE Standard for Distributed Interactive Simulation Application Protocol, (Standard IEEE std 1278.1a-1998) IEEE.

[7] McTaggart, K. (2007), ShipMo3D Version 1.0 User Manual for Simulating Time Domain Motions of a Freely Maneuvering Ship in a Seaway, (DRDC Atlantic Technical Memorandum 2007-172) DRDC Atlantic, Dartmouth, Nova Scotia, Canada.

[8] Canadian Department of National Defence, Seamanship Rigging and Procedures Manual, (Report B-GN-181-105/FP-E00).

[9] Canadian Department of National Defence, Sliding Padeye Receiving Units Bulkhead Mounted (Model SB-9C) and Retractable (Model SR-12C), (Report C-28-463-000/MS-001).

[10] Hepburn Engineering Inc. (2006), Ram Tensioners, (Technical Report Rev. 00).

[11] Torby, B. J. (1984), Advanced Dynamics for Engineers, New York: CBS College Publishing.

Annex A: Federation Input and Output File Descriptions

A.1 Input File Detailed Description

This section describes the details of all input files used by the various federates in the RAS federation. Each federate uses one .ini file that is interpreted by a set of standardized initialization-file parsing functions that are part of Microsoft's registry library. The files must maintain a standardized structure of "application name" and then "key names". A number of federates use additional input files.

DataLogger.ini provides the data logger federate with HLA federation configuration data. In addition it provides the location to which log files will be written. This file can reside in any directory and its' path is given as a parameter to the DataLoggerFed application. Figure A.1 below provides an example of a DataLogger.ini file. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.1) of the specific contents of the file.

ExeMgr.ini provides the execution manager federate with HLA federation configuration data. This file can reside in any directory and its path is given as a parameter to the ExeMgrFed application. Figure A.2 below provides an example of a ExeMgr.ini file. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.2) of the specific content of the file.

RasGear.ini provides the RAS gear federate with HLA federation configuration data. In addition, it assigns the number of RAS cable segments for which the fed-

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=DataLoggerFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   DataLoggerData.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
8. [INTERNAL]
9. OUTPUTDIR=C:/RAS_Federation_Phase2/RAS_Federation/Output_Files/
```

Figure A.1: *Example of DataLogger.ini file.*

Table A.1: *File parameters for DataLogger.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and must always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federates in the RAS federation have to use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FEDERATIONFILE". The value to the right of the key is the file name provided by the user. All federates in the RAS federation must use the same federation file name.
4	Text indicates a key that represents the federate name/type. This text is used by the initialization-file interpreter and must always be "FEDERATETYPE". The value to the right of the key is the federate name provided by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of an additional configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and must always be "OTHERINPUTFILE". The value to the right of the key is the full path of to the additional configuration file. The Data logger federate doesn't use an additional configuration file.
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step value provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key is provided by the user and shouldn't generally be smaller than the logical time step.
8	Text indicates a section internal to the local federate application. This text is used by the initialization-file interpreter and must always be "[INTERNAL]".
9	Text indicates a key that represents a parameter. This text is used by the initialization-file interpreter. In the case of data logger the text is "OUTPUTDIR" and the value to the right is the name of the output directory to where the data logger will log all its output files.

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=ExeMgrFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   ExeMgrData.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
```

Figure A.2: *Example of ExeMgr.ini file.*

Table A.2: *File parameters for ExeMgr.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and must always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federates in the RAS federation must use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FEDERATIONFILE". The value to the right of the key is the file name provided by the user. All federates in the RAS federation must use the same federation file name.
4	Text indicates a key that represents the federate's name/type. This text is used by the initialization file interpreter and must always be "FEDERATETYPE". The value to the right of the key is the federate's name provided by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of an additional configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and must always be "OTHERINPUTFILE". The value to the right of the key is the full path to the additional configuration file. The execution manager federate doesn't use an additional configuration file.
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step value provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key is the LookAhead value provided by the user and shouldn't generally be smaller than the logical time step.

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=RasGearFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   raseqii.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
8. [INTERNAL]
9. NUMOFCABLEPOINTS=8
```

Figure A.3: *Example of RasGear.ini file.*

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=ExeMgrFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   bretSeaState5-330deg.pkl
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
8. [INTERNAL]
9. SEAWAYOPTION=waves
```

Figure A.4: *Example of SeaWay.ini file.*

erate has to publish spatial data. The "OTHERINPUTFILE" parameter points to the input file used by the Fortran portion of the federate. This file can reside in any directory and its path is given as a parameter to the RasGearFed application. Figure A.3 below provides a sample of a RasGear.ini file. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.3) of the specific content of the file.

SeaWay.ini provides the seaway federate with HLA federation configuration data. The "OTHERINPUTFILE" parameter points to an additional file for specific configuration of seaway parameters. This file can reside in any directory and its path is given as a parameter to the SeaWayFed application. Figure A.4 below provides a sample of a SeaWay.ini file. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.4) of the specific content of the file.

Table A.3: *File parameters for RasGear.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and must always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federates in the RAS federation have to use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FEDERATIONFILE". The value to the right of the key is the file name provided by the user. All federates in the RAS federation must use the same federation file name.
4	Text indicates a key that represents the federate's name/type. This text is used by the initialization-file interpreter and must always be "FEDERATETYPE". The value to the right of the key is the federate's name provided by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of a configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and must always be "OTHERINPUTFILE". The value to the right of the key represents the input file used by the Fortran portion of the federate.
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step value provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key provided by the user and shouldn't generally be smaller than the logical time step.
8	Text indicates a section internal to the local federate application. This text is used by the initialization-file interpreter and must always be "[INTERNAL]".
9	Text indicates a key that represents the number of cable segments (number of cable points - 1) for which XYZ positions will be calculated by the federate and sent to data logger. Minimum value is 2 (single segment) and maximum allowed value is 20 (19 segments). This key should always be "NUMOFCABLEPOINTS".

Table A.4: *File parameters for SeaWay.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and must always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federates in the RAS federation must use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FEDERATIONFILE". The value to the right of the key is the file name provided by the user. All federates in the RAS federation must use the same federation file name.
4	Text indicates a key that represents the federate's name/type. This text is used by the initialization-file interpreter and has to always be "FEDERATETYPE". The value to the right of the key is the federate's name provided by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of an additional configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and must always be "OTHERINPUTFILE". The value to the right of the key is the name of the sea way definition file.
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key is a LookAhead value provided by the user and shouldn't generally be smaller than the logical time step.
8	Text indicates a section internal to the local federate application. This text is used by the initialization-file interpreter and must always be "[INTERNAL]".
9	Text indicates a key that represents the state of the sea. This key should always be "SEAWAYOPTION". The value to the right of the key should be "calm" or "waves". The term "waves" indicates that sea way file should be processed. The term "calm" indicates that the seaway file can be ignored.

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=SupplyHelmFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   helmSupplyData.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
```

Figure A.5: *Example of SupplyHelm.ini file.*

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=ReceiveHelmFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   helmReceiveData.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
```

Figure A.6: *Example of ReceiveHelm.ini file.*

SupplyHelm.ini and **ReceiveHelm.ini** provide the supply helm federate and receive helm federate with HLA federation configuration data. This file also provide the name of additional helm control files to be used by the federates. The files can reside in any directory and their path is given as a parameter to the HelmFed application. Figures A.5 and A.6 below provide a sample of a SupplyHelm.ini and ReceiveHelm.ini files. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.5) of the specific content of the file.

Raseqii.inp contains the configuration parameters for the RAS gear computational model such as RAS cable attachment points, payload parameters, integration parameters and some initial conditions. All parameters on a given row are separated by white space (or spaces). This file can reside anywhere and its location is provided by the "OTHERINPUTFILE" parameter in the RasGear.ini file. All parameters are documented in the file and further information is available in Sections 2.4 and 3.4.

SM3DStcx.ini and **SM3DRtcx.ini** provide the supply ship and receive ship federates with HLA configuration data. In addition these files contain the name of the ShipMo3D configuration file to be used by the ship federates. The files can reside

Table A.5: *File parameters for SupplyHelm.ini and ReceiveHelm.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and has to always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federates in the RAS federation must use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FEDERATIONFILE". The value to the right of the key is the files name provided by the user. All federate in the RAS federation have to use the same federation file name.
4	Text indicates a key that represents the federate's name/type. This text is used by the initialization-file interpreter and has to always be "FEDERATETYPE". The value to the right of the key is the federate's name by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of a configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and has to always be "OTHERINPUTFILE". The value to the right of the key represents the input file that provides actual inputs for the helm such as heading, RPM, etc..
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key provided by the user and shouldn't generally be smaller than the logical time step.

```
1.[SM3D]
2.SHIP_INPUT=C:\RAS_Federation_Phase2\SM3D_TestCases\TestCase1\
  ProtecteurDeepFreeMo2.inp
3.DRALGORITHM=4
4.[Federate]
5.SHIP_FUNCTION=S
6.FEDERATION_NAME=RAS
7.FEDERATE_NAME=SupplyShip
8.FED_FILENAME=RAS-FOM.xml
9.TIME_CONSTRAINED=YES
10.TIME_REGULATING=YES
11.TIME_STEP=0.10
12.LOOKAHEAD=0.10
13.RESIGN_ACTION=CANCEL_THEN_DELETE_THEN_DIVEST
14.DEBUG_LEVEL=DEBUG
```

Figure A.7: *Example of SM3DStc1.ini file.*

in any directory and their path is given as a parameter to the sm3dfed python application. Figures A.7 and A.8 below provide an example of a SupplyHelm.ini and ReceiveHelm.ini files. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.6) of the specific content of the file.

The input files **HelmReceiveData.inp** and **HelmSupplyData.inp** provide receiving ship and supply ship with 4 helm input values (from left to right) to be used by the helm federates: propeller RPM, autopilot heading in degrees (where 0.0 represents north), rudder angle in degrees, and helm mode (i.e. whether rudder is controlled based on input command: rudder angle (1) or autopilot (0)). The values are separated by white space (or spaces). This file can reside in any directory since its path is given in ReceiveHelm.ini/SupplyHelm.ini files. The range of value for RPM is limited by the actual ship motion model. The heading and rudder angles should be given in the range of 0.0 to 360.0. Sample input files for HelmReceiveData.inp and HelmSupplyData.inp are provided in Figures A.9 and A.10.

HydrodynamicInteractions.ini provides the hydrodynamic interactions federate with HLA federation configuration data. This file can reside in any directory and its path is given as a parameter to the HydrodynamicInteractionFed application. Figure A.11 below provides an example of a HydrodynamicInteractions.ini file. The first column of numbers indicates the line number in the file and does not form a part of the actual input file data. These line numbers will be used subsequently to provide a detailed description of the meaning (Table A.7) of the specific content of the file.

```
1.[SM3D]
2.SHIP_INPUT=C:\RAS_Federation_Phase2\SM3D_TestCases\TestCase1\
  HalifaxDeepFreeMo2.inp
3.DRALGORITHM=4
4.[Federate]
5.SHIP_FUNCTION=R
6.FEDERATION_NAME=RAS
7.FEDERATE_NAME=ReceiveShip
8.FED_FILENAME=RAS-FOM.xml
9.TIME_CONSTRAINED=YES
10.TIME_REGULATING=YES
11.TIME_STEP=0.10
12.LOOKAHEAD=0.10
13.RESIGN_ACTION=CANCEL_THEN_DELETE_THEN_DIVEST
14.DEBUG_LEVEL=DEBUG
```

Figure A.8: *Example of SM3DRtc1.ini file.*

```
184  0.00  0.0  0
```

Figure A.9: *Example of ReceiveHelm.ini file.*

```
364  0.00  0.0  0
```

Figure A.10: *Example of SupplyHelm.ini file.*

```
1. [FEDERATE]
2. FEDERATIONNAME=RAS
3. FEDERATIONFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   RAS-FOM.xml
4. FEDERATETYPE=HydrodynamicInteractionFed
5. OTHERINPUTFILE=C:/RAS_Federation_Phase2/RAS_Federation/Config/
   HydroulicInt.inp
6. TIME_STEP=0.10
7. LOOKAHEAD=0.10
```

Figure A.11: *Example of HydrodynamicInteractions.ini file.*

Table A.6: *File parameters for SM3DStcx.ini and SM3DRtcx.ini files.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and has to always be "[SM3D]".
2	Text indicates a key that represents an additional configuration file used by ShipMo3D. This key should always be "SHIP_INPUT".
3	Text key for dead reckoning algorithm (enumeration between 1 and 8 as per DIS standard). This key should always be "DRALOGRITHM".
4	Text indicates the federation section. This text is used by the initialization file interpreter and has to always be "[FEDERATE]".
5	Text key for ship's function, which should be "SHIP_FUNCTION". The value to the right should be "R" for receiving ship or "S" for supply ship.
6	Text key for the federation name. This key should always be "FEDERATION_NAME". The value to the right is the federation name given by the user. All federates must use the same federation name.
7	Text indicates a key that represents the federate's name/type. This text is used by the initialization-file interpreter and has to always be "FEDERATE_NAME". The value to the right of the key is the federate's name defined by the user. Each federate must have a unique name.
8	Text indicates a key that represents the federation's file name (also referred as .fed file). This key should always be "FED_FILENAME". The value to the right of the key is the .fed file name provided by the user. All federates in the RAS federation have to use the same federation file name.
9	Text indicates if the federate is time constrained. This key should always be "TIME_CONSTRAINED". The value to the right of the key should be "YES" for proper RAS federation execution.
10	Text indicates if the federate is time regulating. This key should always be "TIME_REGULATING". The value to the right of the key should be "YES" for proper RAS federation execution.
11	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP", followed by logical time step.
12	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key should not generally be smaller than the logical time step.
13	Text indicating the federate's action upon resignation from the federation. This key should always be "RESIGN_ACTION". The value to the right of the key should be "CANCEL_THEN_DELETE_THEN_DIVEST" for proper RAS federation execution.
14	Text indicating debug level. Key should always be "DEBUG_LEVEL". The value right of the key can be "DEBUG" or "INFO".

Table A.7: *File parameters for HydrodynamicInteractions.ini file.*

Line	Input parameters
1	Text indicates the application name. This text is used by the initialization file interpreter and has to always be "[FEDERATE]".
2	Text indicates a key that represents the federation's name. This key should always be "FEDERATIONNAME". The value to the right of the key is a federation name given by the user. All federate in the RAS federation have to use the same federation name.
3	Text indicates a key that represents the federation's file name (also referred as .fed file), this key should always be "FEDERATIONFILE". The value to the right of the key is the file name provided by the user. All federates in the RAS federation have to use the same federation file name.
4	Text indicates a key that represents the federate's name/type. This text is used by the initialization-file interpreter and has to always be "FEDERATETYPE". The value to the right of the key is the federate's name defined by the user. Each federate in the federation must have a unique name.
5	Text indicates a key that represents the name of an additional configuration file (input file) used by that federate. This text is used by the initialization-file interpreter and has to always be "OTHERINPUTFILE". The value to the right of the key is the full path of that additional configuration file. The Hydrodynamic Interaction federate doesn't use an additional configuration file.
6	Text indicates a key that represents the federate's logical time step. This key should always be "TIME_STEP". The value to the right of the key is a logical time step provided by the user.
7	Text indicates a key that represents the federate's LookAhead value. This key should always be "LOOKAHEAD". The value to the right of the key is provided by the user and shouldn't generally be smaller than the logical time step.

```
time Helm-SUPPLY: RPM Heading LinarSpeed RudderAngle Helm-RECEIVE:
RPM Heading LinarSpeed RudderAngle

0.0000000000   0 0.00 0.00   0.00     0 0.00  0.00 0.00
0.1000000000   0 0.00 0.00   0.00     0 0.00  0.00 0.00
0.2000000000 364 0.00 10.30 -1.13   184 0.00 10.30 1.13
0.3000000000 364 0.00 10.30 -1.29   184 0.00 10.30 1.29
0.4000000000  364 0.00 10.30 -1.34   184 0.00 10.30 1.33
```

Figure A.12: *Sample output file.*

A.2 Output File Detailed Description

All output files are generated by the data logger federate. These are ASCII files where each line corresponds to a single time step (that is the first value in each row). The values in each line are separated by a single space and can represent an integer, a float or a double. The data logger will locate the files at a path provided by the DataLogger.ini file. The first line of each log file lists the order of values. A sample (representing just the first 5 time steps) from the helm log file is listed in Figure A.12.

LogHelmData.out This file contains auto-pilot heading input and propeller RPM input as well as actual ship's heading, linear speed, propeller RPM and rudder angle. This single file logs the helm data for both ships.

LogPayLoad.out This file contains 6D spatial data for the pay-load. All values are double precision, floating point values. The 6 left-most values represent the position, the next 6 values represent velocity and the 6 right-most values are the payload's acceleration.

LogCablePoints.out This file contains the 3D position for each of the cable points (cable points are represented by their index and 3D position; indexes are 0 to $N-1$, where N represents the total number of cable points). Cable segment n is positioned between cable point n and cable point $n+1$. The requested number of segments NS ($NS = N-1$) provided to the RAS gear federate as a configuration parameter. Cable segment 0 begins at the supply ship and cable segment $NS-1$ ends at the receiving ship.

Logderask.out This file contains 6D spatial data (position, velocity, acceleration) for both ships. The left-most 18 values are for the supply ship and the next 18 are for the receiving ship.

Logderasf.out This file contains the 3D forces and moment vectors applied to both ships by the RAS gear. In addition this file contains the highline cable tension, the in-haul cable tension and the out-haul cable tension.

This page intentionally left blank.

Annex B: Test Plan

B.1 Introduction

This annex is a high-level overview defining the testing strategy for the High Level Architecture Federation for Ship Replenishment at Sea. Its objective is to communicate project-wide quality standards and procedures. It portrays the state of the project near the end of software development. The testing will concentrate on the behaviour of the overall system with principal focus on the behaviour of the ship and RAS gear federates.

B.2 Objectives

The objective the test plan is to identify and report as many problems with the software system as possible and through this to improve its quality and thus to ultimately satisfactorily verify and validate the software against the requirements identified during the course of this project. Although exhaustive testing is not possible, a reasonably broad range of tests will be performed in order to achieve this goal.

B.3 Testing Strategy

The strategy of the test plan is to demonstrate that the RAS gear simulation performs reasonably under reasonable inputs. The aim is to use the minimum number of tests that will satisfactorily achieve this goal. The client's involvement throughout this process will be sought as much as possible.

B.4 Test Items

The following software components will be used to do the tests (although none will be tested individually through unit testing, at least as part of this test plan):

1. The execution manager federate
2. The seaway federate
3. The supply ship federate
4. The receiving ship federate (same software component as 3)
5. The helm federate for the supply ship
6. The help federate for the receiving ship (same software component as 5)

7. The RAS gear federate
8. The logger federate

B.5 Features to be Tested

The following features will be tested:

1. The behaviour of the supply ship federate in response to the seaway and interaction with the RAS gear federate
2. The behaviour of the receiving ship federate in response to the seaway and interaction with the RAS gear federate
3. The behaviour of the RAS gear federate in response to the supply ship federate and the receiving ship federate

B.6 Features not to be Tested

The following features will not be directly tested:

1. The behaviour of the execution manager federate
2. The behaviour of the seaway federate
3. The behaviour of the logger federate
4. The behaviour of the supply ship helm federate
5. The behaviour of the receiving ship helm federate
6. The behaviour of third party components used by the system
7. System performance
8. System usability
9. System security

B.7 Approach

All test cases will be executed on the same version of the software.

As implied in previous sections, component, integration, conversion, interface, security, recovery, performance, are not in the scope of this test plan. This plan covers acceptance and beta testing only. Regression testing may be done if problems with the software are encountered and corrected.

The testing will involve the entire system, or in HLA terms a federation execution, consisting of the execution manager, seaway, supply ship, receiving ship, supply ship helm, receiving ship helm, RAS gear, and logger federates.

The tests will involve normal operational conditions (i.e. reasonable and usual inputs). The operation of the system under boundary conditions or unreasonable or unusual inputs will not be tested given that the purpose of the testing is to demonstrate that the software can successfully simulate real-life RAS operations rather than the fact that it is "fool proof".

The actual testing will use the black-box approach in which only the inputs to and outputs from the system will be observed.

B.8 Pass/Fail Criteria

The success or failure of the test will be based on whether the overall RAS simulation behaves sensibly and realistically. Most likely this will involve an educated judgment on the part of the tester and whenever possible the results will be compared against behaviours observed during real-life operations.

B.8.1 Suspension Criteria

The tests will be suspended if:

1. The federation consisting of all federates cannot be executed
2. Proper data exchange does not occur
3. Simulation execution takes unreasonably long time
4. Results of the simulation are not sensible

B.8.2 Resumption Criteria

The tests will be resumed after sufficient measures have been taken to address the suspension criteria and there is a reasonable belief of success.

B.8.3 Approval Criteria

The tests results will be approved after they are examined and the behaviour of the simulation, i.e. the ships and the RAS gear, is sensible and realistic.

B.9 Testing Process

The following represents an overview of the steps that will be taken during the testing process:

1. This document will be presented to the client for review and comments. Subsequently, it will be amended to reflect the client's feedback, if any.
2. The specific aspects of the system to be tested will be determined.
3. Several test cases will be identified. Ideally test cases will involve simulations of real-life RAS operations, if documentation for such operations can be obtained.
4. The test cases will be reviewed to ensure that they will adequately test the desired aspects of the system.
5. The expected results for each test will be determined.
6. The test case configuration, test data, and expected results will be collected and documented.
7. The test environment will be set up.
8. The tests will be performed.
9. The tests results will be collected, reviewed and documented.
10. Unsuccessful tests will be used as inputs for fixing problems, and such tests will be repeated until they are successful. Only problems with components developed during the project will be addressed. If any component is changed, all tests will be re-executed.
11. The final test results demonstrating successful execution of the RAS simulation will be presented to the client.

B.9.1 Test Deliverables

The following will comprise the test deliverables for each test case:

1. The document describing each test case, including settings and the steps for its execution

2. The input data including any configuration files
3. Scripts used to automate the test, if available
4. Output data/results
5. Analysis and interpretation of output data/results (e.g. charts, tables, etc)

In addition the deliverables will include the software itself, instructions for its installation and execution, and any other information that may be useful in reproducing the tests.

B.9.2 Testing Tasks

The following tasks are deemed necessary to prepare for and perform testing activities:

1. Identify documentation required to operate third party components, which may be required to perform the tests
2. Document steps required to set up the test platform (write scripts that might automate the process, where reasonable)
3. Configure the test platform
4. Document the steps required to install the RAS federates (write scripts that might automate the process, where reasonable)
5. Install the federation components on the test platform
6. Document the interfaces of the federates (i.e. inputs and outputs, including semantics and syntax)
7. Develop scripts to automate the testing process (i.e. for program execution and evaluation of outputs), where reasonable
8. Run the tests
9. Analyze the results
10. If errors are identified, correct them and re-run the tests
11. Collect and document the results
12. Collect software and data used to perform the tests.

B.9.3 Roles and Responsibilities

This section describes the roles and responsibilities on individuals involved with testing.

The Development Team

The following roles have been identified in the development team.

1) Principal Federation Developer - Dan Bleichman

The principal federation developer is responsible for the development of the software components and configuration data files required to execute the RAS federation, i.e. the development of the execution manager, the seaway, the helm, and the logger federate software components and for integration of the ShipMo3d into the ship federates and the RAS gear computation module into the RAS gear federate, and for creation of configuration data required to operate in the HLA environment.

2) Principal RAS Gear Federate Developer - Rob Langlois

The principal RAS gear federate developer is responsible for the development of the physics-based computation module for modelling the behaviour of the RAS gear.

3) Support Developer - Michael Lichodzijewski

The support developer is responsible for assisting the principal developers as required.

The Test Team

The following roles and responsibilities have been identified for the test team:

1) Principal Test Developer/Evaluator - Rob Langlois

The principal test developer is responsible for identifying, developing, and evaluating test cases.

2) Principal Tester - Dan Bleichman

The principal tester is responsible for implementing and executing test cases.

3) Test Coordinator - Rob Langlois

The test coordinator is responsible for ensuring that the test plan is followed. He is also responsible for collecting and archiving all relevant information.

B.9.4 Schedule

The identification and preparation of test cases has commenced. Testing of the federation will commence once all system components have been deemed developed. Specific dates are yet to be determined, but most likely this will take place in the first half of February 2008.

B.10 Environmental Requirements

This section describes the hardware and software that will be used during the testing process.

B.10.1 Hardware

The following hardware configuration will be used to perform the testing:

1. Pentium 4 at 2.8 GHz
2. 512 MB of RAM
3. 40 GB Hard Drive
4. Super VGA (800 x 600) or higher-resolution video adapter and monitor
5. Keyboard and mouse

B.10.2 Software

The following software configuration will be used to perform the testing:

1. Windows XP Professional, Service Pack 2
2. Mak RTI 1.3 running RTI IEEE 1516
3. ShipMo3d 25-Sep-2007
4. Dislin 9.1 for Python
5. Python 2.4.3
6. Numeric Python 24.2
7. Visual Fortran V9.0 Runtime Libraries
8. Visual C++ V8.0 Runtime Libraries

B.10.3 Tools

Testing may be automated through use of scripts and batch files.

B.10.4 Risks and Assumptions

None identified.

B.11 Change Management Procedures

The following procedure will be used to make changes to this document and to any test cases:

1. A change request will be made to the project manager and will include the reason for the change
2. The project manager will approve or reject the change request
3. An approved change will be logged. A log entry will include the date of the change and the reason for the change.
4. A new version of this document and/or test case will be created. The old version will be saved for future reference.

B.12 Plan Approvals

The project manager will be responsible for approving the test plan and all activities performed during the testing process.

Annex C: Test Cases for the Software Deliverables

This Annex contains descriptions of test cases that will be carried out in accordance with the goals of the Test Plan for the Software Deliverables for ‘ ‘Development of a High Level Architecture Federation for Ship Replenishment at Sea”.

C.1 Test Case 1

C.1.1 Test case number:

1

C.1.2 Test case name:

RAS at midships, calm water (sea state 0), steady speeds, constant RPMs, autopilot heading, typical payload transfer from supply ship to receiving ship.

C.1.3 Test case version:

1

C.1.4 Test created by:

Rob Langlois

C.1.5 Tested by:

Dan Bleichman

C.1.6 Tested on:

March 6, 2008

C.1.7 Test type:

System Level, Black-box

C.1.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate. Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 100 m. The ships are traveling in calm water. Each ship is traveling at a speed of 20 knots. Both are set to travel at a course of 0.0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 20 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 5 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.1.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.1.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
    a. Ship model file = ProtecteurDeepFreeShip.pkl
    b. Seaway option = calm
    c. Seaway model file = N/A
    d. Time step = 0.1s
    e. Initial time = 0s
    f. End of ramp wave function = 20s
    g. Beginning of statistics sampling = 20s

h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = 0m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 20 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 364

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
   a. Ship model file = HalifaxDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = -100m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 20 Knots
   p. Initial rudder deflections = 0 Degrees
   q. Initial rudder speeds = 0
   r. Initial propeller RPMs = 184

3) ShipMo3d model of the seaway - N/A calm sea

4) The supply ship federate configuration file - SM3DStc1.ini
    a. Ship model file = ProtecteurDeepFreeMo2.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = RID.mtl
5) The receiving ship federate configuration file - SM3DRtc1.ini
    a. Ship model file = HalifaxDeepFreeMo2.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. RPM = 364
    b. Autopilot Heading = 0
    c. The helm input data file = helmSupplyData.inp
    d. FOM file = RAS-FOM.xml
    e. RID file = rid.mtl
7) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. RPM = 184
    b. Autopilot Heading = 0
    c. The helm input data file = helmReceiveData.inp
    d. FOM file = RAS-FOM.xml
    e. RID file = rid.mtl
8) The seaway federate configuration file - SeaWay.ini
    a. N/A
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output file = Logger.out
10) The RAS gear federate configuration file - RasGear.ini

a. RAS gear computation engine configuration file = Raseqii.inp (rename Case1_raseqii.inp)

b. FOM file = RAS-FOM.xml

c. RID file = rid.mtl

11) The RAS gear computation engine configuration file - Raseqii.inp

a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)

b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)

c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)

e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)

f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)

g. Payload present = yes

h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)

i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)

j. Payload mass = 1000 kg

k. All other parameters have been set to default values as documented in Section D.

12) The execution manager configuration file - ExeMgr.ini

a. FOM file = RAS-FOM.xml

b. RID file = rid.mtl

13) The RID file - rid.mtl (constant for all tests; see references)

14) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.1.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.1.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running

4) Open 8 command prompts

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc1.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc1.ini`), but do not yet press Enter.

12) At the eighths command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config \RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one is launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.1.13 Outputs:

C.1.14 Interpretation of results:

C.1.15 Pass/Fail:

C.1.16 Approval:

C.2 Test Case 2

C.2.1 Test case number:

2

C.2.2 Test case name:

RAS near the bow, calm water (sea state 0), steady speeds, constant RPMs, autopilot heading, typical payload transfer from supply ship to receiving ship.

C.2.3 Test case version:

1

C.2.4 Test created by:

Rob Langlois

C.2.5 Tested by:

Dan Bleichman

C.2.6 Tested on:

March 6, 2008

C.2.7 Test type:

System Level, Black-box

C.2.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 100 m. The ships are traveling in calm water. Each ship is traveling at a speed of 20 knots. Both are set to travel at a course of 0.0 deg (straight north) on

autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 20 knots. The mass of the crate is 1000 Kg. The RAS gear is located near bow on both ships. On the HMCS Protecteur it is near bow and at elevation of 18 m above the waterline. On the HMCS Halifax it near bow and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.2.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.2.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 20 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 364

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
   a. Ship model file = HalifaxDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = -100m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 20 Knots
   p. Initial rudder deflections = 0 Degrees
   q. Initial rudder speeds = 0
   r. Initial propeller RPMs = 184

3) ShipMo3d model of the seaway - N/A calm sea

4) The supply ship federate configuration file - SM3DStc1.ini
   a. Ship model file = ProtecteurDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml
   c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc1.ini
   a. Ship model file = HalifaxDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml

c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
   a. RPM = 364
   b. Autopilot Heading = 0
   c. The helm input data file = helmSupplyData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

7) The receiving ship helm federate configuration file - ReceivingHelm.ini
   a. RPM = 184
   b. Autopilot Heading = 0
   c. The helm input data file = helmReceiveData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

8) The seaway federate configuration file - SeaWay.ini
   a. N/A
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

9) The logger federate configuration file - dataLogger.ini
   a. FOM file = RAS-FOM.xml
   b. RID file = rid.mtl
   c. Output file = Logger.out

10) The RAS gear federate configuration file - RasGear.ini
   a. RAS gear computation engine configuration file = Raseqii.inp (rename Case2_raseqii.inp)
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

11) The RAS gear computation engine configuration file - Raseqii.inp
   a. Supply ship's highline cable's attachment point (m) = (50, -4.75, 18)

b. Supply ship's in-hull cable's attachment point (m) = (50, -5, 17.5)

c. Supply ship's out-hull cable's attachment point (m) = (50, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (50, 4.75, 15)

e. Receiving ship's in-hull cable's attachment point = (50, 5, 14.75)

f. Receiving ship's out-hull cable's attachment point = (50, 5, 15.25)

g. Payload present = yes

h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)

i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)

j. Payload mass = 1000 kg

k. All other parameters have been set to default values as documented in Annex B.

12) The execution manager configuration file - ExeMgr.ini

a. FOM file = RAS-FOM.xml

b. RID file = rid.mtl

13) The RID file - rid.mtl (constant for all tests; see references)

14) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.2.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.2.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running.

4) Open 8 command prompts.

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc1.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc1.ini`), but do not yet press Enter.

12) At the eighths command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config \RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one is launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.2.13 Outputs:

C.2.14 Interpretation of results:

C.2.15 Pass/Fail:

C.2.16 Approval:

C.3 Test Case 3

C.3.1 Test case number:

3

C.3.2 Test case name:

RAS near the stern, calm water (sea state 0), steady speeds, constant RPMs, autopilot heading, typical payload transfer from supply ship to receiving ship.

C.3.3 Test case version:

1

C.3.4 Test created by:

Rob Langlois

C.3.5 Tested by:

Dan Bleichman

C.3.6 Tested on:

March 6, 2008

C.3.7 Test type:

System Level, Black-box

C.3.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 100 m. The ships are traveling in calm water. Each ship is traveling at a speed of 20 knots. Both are set to travel at a course of 0.0 deg (straight north) on

autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 20 knots. The mass of the crate is 1000 Kg. The RAS gear is located near the stern on both ships. On the HMCS Protecteur it at stern and at elevation of 18 m above the waterline. On the HMCS Halifax it is at stern and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.3.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)
2) The receiving ship federate (instance of the ship motions federate software component)
3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.3.10 Input:

1) ShipMo3D model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 20 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 364

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
   a. Ship model file = HalifaxDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = -100m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 20 Knots
   p. Initial rudder deflections = 0 Degrees
   q. Initial rudder speeds = 0
   r. Initial propeller RPMs = 184

3) ShipMo3d model of the seaway - N/A calm sea

4) The supply ship federate configuration file - SM3DStc1.ini
   a. Ship model file = ProtecteurDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml
   c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc1.ini
   a. Ship model file = HalifaxDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml

c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
   a. RPM = 364
   b. Autopilot Heading = 0
   c. The helm input data file = helmSupplyData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

7) The receiving ship helm federate configuration file - ReceivingHelm.ini
   a. RPM = 184
   b. Autopilot Heading = 0
   c. The helm input data file = helmReceiveData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

8) The seaway federate configuration file - SeaWay.ini
   a. N/A
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

9) The logger federate configuration file - dataLogger.ini
   a. FOM file = RAS-FOM.xml
   b. RID file = rid.mtl
   c. Output file = Logger.out

10) The RAS gear federate configuration file - RasGear.ini
   a. RAS gear computation engine configuration file = Raseqii.inp (rename Case3_raseqii.inp)
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

11) The RAS gear computation engine configuration file - Raseqii.inp
   a. Supply ship's highline cable's attachment point (m) = (-50, -4.75, 18)
   b. Supply ship's in-hull cable's attachment point (m) = (-50, -5, 17.5)

c. Supply ship's out-hull cable's attachment point (m) = (-50, -5, 18.5)
d. Receiving ship's highline cable's attachment point = (-50, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (-50, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (-50, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

12) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
13) The RID file - rid.mtl (constant for all tests; see references)
14) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.3.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.3.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.
2) Prepare configuration files specific to this test.
3) Make sure that license server is running
4) Open 8 command prompts
5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\ sm3dfed1.py -c SM3DStc1.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\ sm3dfed1.py -c SM3DRtc1.ini`), but do not yet press Enter.

12) At the eighths command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C\:\RAS_Federation_Phase3\Config \RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.3.13 Outputs:

C.3.14 Interpretation of results:

C.3.15 Pass/Fail:

C.3.16 Approval:

C.4 Test Case 4

C.4.1 Test case number:

4

C.4.2 Test case name:

RAS at midships, oblique seas, small waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.4.3 Test case version:

2

C.4.4 Test created by:

Rob Langlois

C.4.5 Tested by:

Dan Bleichman

C.4.6 Tested on:

January 21, 2009

C.4.7 Test type:

System Level, Black-box

C.4.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in sea state 2, with waves from 315 degrees. Each ship is traveling at an initial speed of 20 knots. Both are set to travel

at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.4.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.4.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = bretSeaState2-315deg.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = bretSeaState2-315deg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -100m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 108.67

3) ShipMo3d model of the seaway - bretSeaState2-315deg.pkl

4) The supply ship federate configuration file - SM3DStc4.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc4.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 109
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = bretSeaState2-315deg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case4_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl

15) The RID file - rid.mtl (constant for all tests; see references)

16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.4.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.4.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running

4) Open 8 command prompts

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc4.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc4.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.4.13 Outputs:

C.4.14 Interpretation of results:

C.4.15 Pass/Fail:

C.4.16 Approval:

C.5 Test Case 5

C.5.1 Test case number:

5

C.5.2 Test case name:

RAS at midships, oblique seas, large waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.5.3 Test case version:

2

C.5.4 Test created by:

Rob Langlois

C.5.5 Tested by:

Dan Bleichman

C.5.6 Tested on:

January 21, 2009

C.5.7 Test type:

System Level, Black-box

C.5.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in sea state 4, with waves from 315 degrees. Each ship is traveling at an initial speed of 20 knots. Both are set to travel

at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.5.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.5.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = bretSeaState4-315deg.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = bretSeaState4-315deg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -100m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 20 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 108.67

3) ShipMo3d model of the seaway - bretSeaState4-315deg.pkl

4) The supply ship federate configuration file - SM3DStc5.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc5.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 109
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = bretSeaState4-315deg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case5_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
15) The RID file - rid.mtl (constant for all tests; see references)
16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.5.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.5.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.
2) Prepare configuration files specific to this test.
3) Make sure that license server is running
4) Open 8 command prompts
5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc5.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc5.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.5.13 Outputs:

C.5.14 Interpretation of results:

C.5.15 Pass/Fail:

C.5.16 Approval:

C.6 Test Case 6

C.6.1 Test case number:

6

C.6.2 Test case name:

RAS at midships, calm water (sea state 0), steady speeds, constant RPMs, autopilot heading, typical payload transfer from supply ship to receiving ship.

C.6.3 Test case version:

1

C.6.4 Test created by:

Rob Langlois

C.6.5 Tested by:

Dan Bleichman

C.6.6 Tested on:

March 6, 2008

C.6.7 Test type:

System Level, Black-box

C.6.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur, the supply ship, attached to HMCS Halifax, the receiving ship via the RAS gear and cables without a payload. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 100 m. The ships are traveling in calm water. Each ship is traveling at a speed of

20 knots. Both are set to travel at a course of 0.0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 20 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 5 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 5 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.6.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.6.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = calm
   c. Seaway model file = N/A
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees

o. Initial speed = 20 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 364

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = calm
c. Seaway model file = N/A
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -100m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 20 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 184

3) ShipMo3d model of the seaway - N/A calm sea

4) The supply ship federate configuration file - SM3DStc1.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc1.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml

c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
   a. RPM = 364
   b. Autopilot Heading = 0
   c. The helm input data file = helmSupplyData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

7) The receiving ship helm federate configuration file - ReceivingHelm.ini
   a. RPM = 184
   b. Autopilot Heading = 0
   c. The helm input data file = helmReceiveData.inp
   d. FOM file = RAS-FOM.xml
   e. RID file = rid.mtl

8) The seaway federate configuration file - SeaWay.ini
   a. N/A
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

9) The logger federate configuration file - dataLogger.ini
   a. FOM file = RAS-FOM.xml
   b. RID file = rid.mtl
   c. Output file = Logger.out

10) The RAS gear federate configuration file - RasGear.ini
   a. RAS gear computation engine configuration file = Raseqii.inp (rename Case6_raseqii.inp)
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

11) The RAS gear computation engine configuration file - Raseqii.inp
   a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
   b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
   c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)
   d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)

e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)

f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)

g. Payload present = No (plflag set to 0)

h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)

i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)

j. Payload mass = 1000 kg

k. All other parameters have been set to default values as documented in Section D.

12) The execution manager configuration file - ExeMgr.ini

a. FOM file = RAS-FOM.xml

b. RID file = rid.mtl

13) The RID file - rid.mtl (constant for all tests; see references)

14) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.6.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.6.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running

4) Open 8 command prompts

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\ sm3dfed1.py -c SM3DStc1.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\ sm3dfed1.py -c SM3DRtc1.ini`), but do not yet press Enter.

12) At the eighths command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe` C`:\RAS_Federation_Phase3\Config \RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.6.13 Outputs:

C.6.14 Interpretation of results:

C.6.15 Pass/Fail:

C.6.16 Approval:

C.7 Test Case 7

C.7.1 Test case number:

7

C.7.2 Test case name:

RAS at midships, head seas, large regular waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.7.3 Test case version:

1

C.7.4 Test created by:

Rob Langlois

C.7.5 Tested by:

Dan Bleichman

C.7.6 Tested on:

November 21, 2008

C.7.7 Test type:

System Level, Black-box

C.7.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in regular waves at sea state 5 coming from a heading of 0 degrees. Each ship is traveling at a speed of 12 knots. Both are set to

travel at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.7.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.7.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = regSeawayH5ZeroDeg.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = regSeawayH5ZeroDeg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -52.25m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 110.67

3) ShipMo3d model of the seaway - regSeawayH5ZeroDeg.pkl

4) The supply ship federate configuration file - SM3DStc7.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc7.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 111
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = regSeawayH5ZeroDeg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case7_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. The execution manager input data file = ExeMgrData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
15) The RID file - rid.mtl (constant for all tests; see references)
16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.7.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.7.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.
2) Prepare configuration files specific to this test.
3) Make sure that license server is running
4) Open 8 command prompts
5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc7.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc7.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.7.13 Outputs:

C.7.14 Interpretation of results:

C.7.15 Pass/Fail:

C.7.16 Approval:

C.8 Test Case 8

C.8.1 Test case number:

8

C.8.2 Test case name:

RAS at midships, head seas, large irregular waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.8.3 Test case version:

1

C.8.4 Test created by:

Rob Langlois

C.8.5 Tested by:

Dan Bleichman

C.8.6 Tested on:

November 26, 2008

C.8.7 Test type:

System Level, Black-box

C.8.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in irregular waves at sea state 5 coming from a heading of 0 degrees. Each ship is traveling at a speed of 12 knots. Both are set to

travel at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.8.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.8.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = bretSeaState5.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = bretSeaState5.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -52.25m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 110.67

3) ShipMo3d model of the seaway - bretSeaState5.pkl

4) The supply ship federate configuration file - SM3DStc8.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc8.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 111
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = bretSeaState5.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case8_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl

15) The RID file - rid.mtl (constant for all tests; see references)

16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.8.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.8.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running

4) Open 8 command prompts

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc8.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc8.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one os launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.8.13 Outputs:

C.8.14 Interpretation of results:

C.8.15 Pass/Fail:

C.8.16 Approval:

C.9 Test Case 9

C.9.1 Test case number:

9

C.9.2 Test case name:

RAS at midships, oblique seas, large regular waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.9.3 Test case version:

1

C.9.4 Test created by:

Rob Langlois

C.9.5 Tested by:

Dan Bleichman

C.9.6 Tested on:

November 21, 2008

C.9.7 Test type:

System Level, Black-box

C.9.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in regular waves at sea state 5 coming from a heading of 330 degrees. Each ship is traveling at a speed of 12 knots. Both are set

to travel at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.9.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.9.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = regSeawayH5330Deg.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = regSeawayH5330Deg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -52.25m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 110.67

3) ShipMo3d model of the seaway - regSeawayH5330Deg.pkl

4) The supply ship federate configuration file - SM3DStc9.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc9.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 111
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = regSeawayH5330Deg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case9_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
15) The RID file - rid.mtl (constant for all tests; see references)
16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.9.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.9.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.
2) Prepare configuration files specific to this test.
3) Make sure that license server is running
4) Open 8 command prompts
5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc9.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc9.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one is launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, then stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.9.13 Outputs:

C.9.14 Interpretation of results:

C.9.15 Pass/Fail:

C.9.16 Approval:

C.10 Test Case 10

C.10.1 Test case number:

10

C.10.2 Test case name:

RAS at midships, oblique seas, large irregular waves, steady speeds, constant RPMs, zero autopilot heading, typical payload transfer from supply ship to receiving ship.

C.10.3 Test case version:

1

C.10.4 Test created by:

Rob Langlois

C.10.5 Tested by:

Dan Bleichman

C.10.6 Tested on:

January 7, 2009

C.10.7 Test type:

System Level, Black-box

C.10.8 Test case description:

The purpose of this test is to ensure that the RAS federation generates sensible behaviors for the supply ship federate, the receiving ship federate, and the RAS gear federate.

Specifically, this test simulates a RAS operation involving the HMCS Protecteur supplying a crate to HMCS Halifax. The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in irregular waves at sea state 5 coming from a heading of 330 degrees. Each ship is traveling at a speed of 12 knots. Both are set

to travel at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear is using standard settings (see RAS gear federate reference manual). The simulation runs for 80s where payload transfer starts 40s into the simulation (to insure steady ship motion conditions).

C.10.9 Items to be tested:

1) The supply ship federate (instance of the ship motions federate software component)

2) The receiving ship federate (instance of the ship motions federate software component)

3) The RAS gear federate (instance of the RAS gear computation model federate software component)

C.10.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
   a. Ship model file = ProtecteurDeepFreeShip.pkl
   b. Seaway option = waves
   c. Seaway model file = regSeawayH5-330deg.pkl
   d. Time step = 0.1s
   e. Initial time = 0s
   f. End of ramp wave function = 20s
   g. Beginning of statistics sampling = 20s
   h. Non-linear option = Linear
   i. Initial x-coordinate = 0m
   j. Initial y-coordinate = 0m
   k. Initial heave = 0 m
   l. Initial roll = 0 Degrees
   m. Initial pitch = 0 Degrees
   n. Initial heading = 0 Degrees
   o. Initial speed = 12 Knots

p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 218.5

2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp
a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = regSeawayH5-330deg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -52.25m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 110.67

3) ShipMo3d model of the seaway - regSeawayH5-330deg.pkl

4) The supply ship federate configuration file - SM3DStc10.ini
a. Ship model file = ProtecteurDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc10.ini
a. Ship model file = HalifaxDeepFreeMo2.inp
b. FOM file = RAS-FOM.xml
c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
    a. The helm input data file = helmSupplyData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 111
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = regSeawayH5-330deg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case10_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)

d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
g. Payload present = yes
h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)
i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)
j. Payload mass = 1000 kg
k. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
15) The RID file - rid.mtl (constant for all tests; see references)
16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.10.11 Expected output:

The lateral distance between ships should decrease slightly due to the RAS gear forces. The payload should initialize aligned with the supply ship, should traverse to the receiving ship consistent with the commanded traverse speed profile and traverse system limits specified in the raseqii.inp input file, and should stop at the receiving ship attachment point. The payload body should show dynamic behaviour consistent with the ship motions.

C.10.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.
2) Prepare configuration files specific to this test.
3) Make sure that license server is running
4) Open 8 command prompts
5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc10.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc10.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one is launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the

receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

C.10.13 Outputs:

C.10.14 Interpretation of results:

C.10.15 Pass/Fail:

C.10.16 Approval:

C.11 Test Case 11

C.11.1 Test case number:

11

C.11.2 Test case name:

RAS at midships, oblique seas, large irregular waves, steady speeds, constant RPMs, zero autopilot heading, no coupling between ships.

C.11.3 Test case version:

1

C.11.4 Test created by:

Rob Langlois

C.11.5 Tested by:

Dan Bleichman

C.11.6 Tested on:

March 12, 2009

C.11.7 Test type:

System Level, Black-box

C.11.8 Test case description:

The main goal of the eleventh test case was to verify, that when running the federation without the mechanical coupling provided by the RAS gear, the ship motions generated when operating ShipMo3D within the federation are identical to those generated when running the simulation outside of the federation using SM3DFreeMo.

The HMCS Protecteur is on the port side of HMCS Halifax. The ships are aligned at mid-ships and parallel centerlines separated by 52.25 m. The ships are traveling in irregular waves at sea state 5 coming from a heading of 330 degrees. Each ship is

traveling at a speed of 12 knots. Both are set to travel at a course of 0 deg (straight north) on autopilot heading. The propellers on both are running at sufficient RPM to generate a speed of 12 knots. The mass of the crate is 1000 Kg. The RAS gear is located at mid-ships on both ships. On the HMCS Protecteur it is at 4.75 m starboard of centerline and at elevation of 18 m above the waterline. On the HMCS Halifax it is at 4.75 m port of centerline and at elevation of 15 m above baseline. The RAS gear kinematic and kinetic output is set to zero (no mechanical coupling). The simulation runs for 80s.

C.11.9 Items to be tested:

1) Ship motions produced by the supply ship federate.

C.11.10 Input:

1) ShipMo3d model of the HMCS Protecteur - ProtecteurDeepFreeMo2.inp
    a. Ship model file = ProtecteurDeepFreeShip.pkl
    b. Seaway option = waves
    c. Seaway model file = regSeawayH5-330deg.pkl
    d. Time step = 0.1s
    e. Initial time = 0s
    f. End of ramp wave function = 20s
    g. Beginning of statistics sampling = 20s
    h. Non-linear option = Linear
    i. Initial x-coordinate = 0m
    j. Initial y-coordinate = 0m
    k. Initial heave = 0 m
    l. Initial roll = 0 Degrees
    m. Initial pitch = 0 Degrees
    n. Initial heading = 0 Degrees
    o. Initial speed = 12 Knots
    p. Initial rudder deflections = 0 Degrees
    q. Initial rudder speeds = 0
    r. Initial propeller RPMs = 218.5
2) ShipMo3d model of the HMCS Halifax - HalifaxDeepFreeMo2.inp

a. Ship model file = HalifaxDeepFreeShip.pkl
b. Seaway option = waves
c. Seaway model file = regSeawayH5-330deg.pkl
d. Time step = 0.1s
e. Initial time = 0s
f. End of ramp wave function = 20s
g. Beginning of statistics sampling = 20s
h. Non-linear option = Linear
i. Initial x-coordinate = 0m
j. Initial y-coordinate = -52.25m
k. Initial heave = 0 m
l. Initial roll = 0 Degrees
m. Initial pitch = 0 Degrees
n. Initial heading = 0 Degrees
o. Initial speed = 12 Knots
p. Initial rudder deflections = 0 Degrees
q. Initial rudder speeds = 0
r. Initial propeller RPMs = 110.67

3) ShipMo3d model of the seaway - regSeawayH5-330deg.pkl

4) The supply ship federate configuration file - SM3DStc10.ini
   a. Ship model file = ProtecteurDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml
   c. RID file = RID.mtl

5) The receiving ship federate configuration file - SM3DRtc10.ini
   a. Ship model file = HalifaxDeepFreeMo2.inp
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

6) The supply ship helm federate configuration file - SupplyHelm.ini
   a. The helm input data file = helmSupplyData.inp
   b. FOM file = RAS-FOM.xml
   c. RID file = rid.mtl

7) The helm supply data file - helpSupplyData.inp
    a. RPM = 218
    b. Autopilot Heading = 0
8) The receiving ship helm federate configuration file - ReceivingHelm.ini
    a. The helm input data file = helmReceiveData.inp
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
9) The helm supply data file - helmReceiveData.inp
    a. RPM = 111
    b. Autopilot Heading = 0
10) The seaway federate configuration file - SeaWay.ini
    a. Seaway model file = regSeawayH5-330deg.pkl
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
11) The logger federate configuration file - dataLogger.ini
    a. FOM file = RAS-FOM.xml
    b. RID file = rid.mtl
    c. Output folder =
12) The RAS gear federate configuration file - RasGear.ini
    a. RAS gear computation engine configuration file = Raseqii.inp (rename Case10_raseqii.inp)
    b. FOM file = RAS-FOM.xml
    c. RID file = rid.mtl
13) The RAS gear computation engine configuration file - Raseqii.inp
    a. Supply ship's highline cable's attachment point (m) = (0, -4.75, 18)
    b. Supply ship's in-hull cable's attachment point (m) = (0, -5, 17.5)
    c. Supply ship's out-hull cable's attachment point (m) = (0, -5, 18.5)
    d. Receiving ship's highline cable's attachment point = (0, 4.75, 15)
    e. Receiving ship's in-hull cable's attachment point = (0, 5, 14.75)
    f. Receiving ship's out-hull cable's attachment point = (0, 5, 15.25)
    g. Payload present = yes

h. Payload/highline attachment point (pkpsh) = (0, 0, 2.8702)

i. Payload/in-out-hull attachment point (plpsh) = (0, 0, 2.6202)

j. Payload mass = 1000 kg

k. Integration parameter "output flag" set to 2

l. Integration parameter "tStartForces" set to 80

m. All other parameters have been set to default values as documented in Section D.

14) The execution manager configuration file - ExeMgr.ini

a. FOM file = RAS-FOM.xml

b. RID file = rid.mtl

15) The RID file - rid.mtl (constant for all tests; see references)

16) The FOM file - RAS-FOM.xml (constant for all tests; see references)

C.11.11 Expected output:

The ship motions produced by the federation should match the motions produced by running SM3DFreeMo outside the federation.

C.11.12 Procedural steps:

1) Make sure that the federation and third party components have been properly installed and configured.

2) Prepare configuration files specific to this test.

3) Make sure that license server is running

4) Open 8 command prompts

5) At the first command prompt type the path of the MAK RTI1516 executable (e.g. `C:\MAK\makRti3.1.1\bin\rtiexec1516.exe`) and then press Enter to launch it.

6) At the second command prompt type in the path of the logger federate executable (e.g. `C:\RAS_Federation_Phase3\DataLoggerFed.exe C:\RAS_Federation_Phase3\Config\DataLogger.ini`), but do not yet press Enter.

7) At the third command prompt type in the path of the execution manager executable (e.g. `C:\RAS_Federation_Phase3\ExeMgrFed.exe C:\RAS_Federation_Phase3\Config\ExeMgr.ini`), but do not yet press Enter.

8) At the forth command prompt type in the path of the supply ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\SupplyHelm.ini`), but do not yet press Enter.

9) At the Fifth command prompt type in the path of the receiving ship helm federate executable followed by the name of the corresponding configuration file (e.g. `C:\RAS_Federation_Phase3\HelmFed.exe C:\RAS_Federation_Phase3\Config\ReceiveHelm.ini`), but do not yet press Enter

10) At the Sixth command prompt type in the "python" command followed by path of the supply ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DStc10.ini`), but do not yet press Enter.

11) At the Seventh command prompt type in the "python" command followed by the path of the receiving ship federate executable followed by the name of the corresponding configuration file (e.g. `python C:\RAS_Federation_Phase3\sm3dfed1.py -c SM3DRtc10.ini`), but do not yet press Enter.

12) At the Eighth command prompt type in the path of the RAS gear federate executable followed (e.g. `C:\RAS_Federation_Phase3\RASGearFed.exe C:\RAS_Federation_Phase3\Config\RasGear.ini`), but do not yet press Enter.

13) Launch in sequence the all 7 federates (you have 60 seconds to do so between the time the first one is launched and the 7th one is launched) Wait for the simulation to end. - Wait for 80 seconds of execution - reported on the ExeMgrFed console, than stop the simulation by hitting the "q" key.

14) Examine the federation log files by plotting ship and payload kinematic data found in the output files `Logderask.out` and `LogPayLoad.out`. Determine the change in lateral distance between the RAS gear on the supply ship and the receiving ship over time. (The data in the output files if formatted and may be imported into Microsoft Excel.)

15) To execute SM3D outside the federation, use the following python code:

```
simtime = 0.0

while (simtime < 80.0) :
    simtime = simtime + 0.1
    # Update the rmp as requested by the helm
    self.freeShipInSeawayTD.setPropellerCommandRPMMode(-1, 218.0)
    # Update the autopilot heading as requested by the helm (deg)
    self.freeShipInSeawayTD.setRudderCommandHeadingMode(-1, 0.0)
    self.freeShipInSeawayTD.setRudderCommandHeadingMode(0, 0.0)
    # Do the time-step
    self.freeShipInSeawayTD.advanceTime(simtime)
    # Post the actual motions, i.e. displacements (m),
    # velocities (m/s), and accelerations (m/s^2)
    # in fixed-earth coordinate system
    [self.dispsFixedTMRad, self.velsFixedTMRad, self.accsFixedTMRad] =
    self.freeShipInSeawayTD.getMotions(self.simtime)
    # Post the actual rpm values for each propeller
    self.shipRPM = self.freeShipInSeawayTD.rpmsPropellers[-1]
    # Post the actual heading (deg) in fixed-earth coordinate system -
    # convert as necessary
    self.shipHeading =
        57.29*self.freeShipInSeawayTD.dispsFixedMRad[-1][5]
    # Post the actual speed (m/s) along instaneous heading - convert
    # as necessary
    self.shipSpeed = self.freeShipInSeawayTD.speedsShipFixed[-1]
    # Post the actual rudder angles (deg) - check the
    # coordinate system used and convert as necessary
    self.ShipRudderAngle =
        57.29*self.freeShipInSeawayTD.rudderDeflectsRad[-1][0]
    print simtime, self.dispsFixedTMRad[0], self.dispsFixedTMRad[1],
          self.dispsFixedTMRad[2], self.dispsFixedTMRad[3],
          self.dispsFixedTMRad[4], self.dispsFixedTMRad[5]
```

16) Initialize SM3D including reading all .pkl definition files as was given to me by DRDC Atlantic

17) Execute SM3D for 80 seconds using the following python code (note that RPM and autopilot heading are hardcoded)

C.11.13 Outputs:

C.11.14 Interpretation of results:

C.11.15 Pass/Fail:

C.11.16 Approval:

Annex D: Reference RASEQII.INP File Used for Testing

```
Coordinate systems:
2                               Euler angle sequence used for incoming ship motion and payload initial conditions (1=XYZ;2=ZYX)
Supply ship attachment points:
  0.     -4.75    18.0         highline attachment point
  0.     -5.      17.5         in-haul attachment point
  0.     -5.      18.5         out-haul attachment point
Receiving ship attachment points:
 40.      4.75    15.          highline attachment point
 40.      5.      14.75        in-haul attachment point
 40.      5.      15.25        out-haul attachment point
Payload parameters:
1                               plflag (0=highline only;1=payload present)
0.        0.       2.8702      pkpsh(x,y,z)
0.        0.       2.6202      plpsh(x,y,z)
1000.                           mass
 670.     0.       0.          mass moment of inertia matrix (Ixx,Ixy,Ixz)
        540.       0.          mass moment of inertia matrix (    Ixy,Ixz)
                   860.        mass moment of inertia matrix (        Ixz)
250.  250.   250.               payload translational viscous damping coefficients
025.  025.   025.               payload rotational viscous damping coefficients
Ram tensioner parameters:
12.    10.    24.               wire rope run lengths (Lad,Ldf,Lr0)
1.379d7  0.180856  6            ram nominal pressure, cylinder diameter, number of cable falls
0.05  1.0                       highline tension friction factor, cable velocity threshold for full frictional tension
1.693557                        nominal system air volume
1.5   0.1                       ram deadband length qramdb, qramtol
0     1.d6                      flag enabling cable stretch (0=off,1=on),AE of the highline cable
1.9   2.0   1200000.           extension bump stop contact distance, ram extension limit, bump spring stiffness
8.333 0.0                       winch speed gain cwinch (m/s/m), winch speed integral gain ki
3.81  71168.  93000.            q13dmax, Thlmax, available highline winch power
In-/out-haul system parameters:
7                               number of command points (maximum=20)
0.00   0.20   0.80  0.85  0.90  0.95  1.00  ratio of payload position
0.8    0.8    0.8   0.8   0.4   0.2   0.1   ratio of maximum traverse speed commanded
 4450.      6.1                 nominal in-/out-haul cable pretension, in-/out-haul cable design speed
50000.  50000.                  winch controller gains between tension and velocity error (in-haul,out-haul)
26700.  26700.                  maximum cable tension before slip (in-haul,out-haul)
    6.1     6.1                 maximum winch cable take-up rates (in-haul,out-haul)
93000.  93000.                  maximum winch power (in-haul,out-haul)
Integration parameters:
0                               flag to produce null output (1=zero outputs;other=run full model)
9.81                            acceleration due to gravity (consistent with Imperial or metric units)
0.01    10                      time step,number of internal time steps per output time step
Initial conditions:
0.                              qram0
0.                              accumulated integral error in RAM position
2                               coordinate system in which payload ICs are specified (2=supply ship)
0.       -5.1      18.          relative position vector to highline/payload interface point
0.        0.        0.          Euler angles of payload coordinate system relative to reference coordinate system
0.        0.        0.          relative velocity vector to highline/payload interface point from reference coordinate system origin
0.        0.        0.          angular velocity of payload expressed in the reference coordinate system
```

This page intentionally left blank.

Distribution List

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

Dan Bleichman, Rob Langlois, Michael Lichodzijewski, Dave Brennan; DRDC Atlantic CR 2011-261; Defence R&D Canada – Atlantic; October 2011.

LIST PART 1: Internal Distribution by Centre

1 Author (1 hard copy)

3 DRDC Atlantic Library (1 hard copy)

3 Project Officer ABCA-02-01 (CSci) for distribution

12 Project Officer ABCANZ-97-12 (H/WP) for distribution

19 TOTAL LIST PART 1 (17 CDs, 2 Hard Copies)

LIST PART 2: External Distribution by DRDKIM

1 Library and Archives Canada

1 DRDKIM 3

1 Mark Tunnicliffe, DSTM 7
National Defence Headquarters
Major-General George R. Pearkes Building
101 Colonel By Drive
Ottawa, ON K1A 0K2

1 DMSS 2
555 Blvd. De la Carriere
Gatineau, QC K1A 0K2

1 Canadian Forces
Maritime Warfare School
Attention: Commanding Officer
PO Box 99000 STN Forces,
Halifax, NS B3K 5X5

1 Director General (1 hard copy)
Defence R&D Canada – Suffield
P.O. Box 4000
Medicine Hat, AB T1A 8K6

1 Director-General
Institute for Ocean Technology
National Research Council of Canada
P.O. Box 12093, Station A
St. John's, NL A1B 3T5

1 Transport Canada / Marine Safety
Tower C, Place de Ville
330 Sparks Street, 11th Floor
Ottawa, ON K1A ON8

2 Commanding Officer (2 hard copies)
U.S. Coast Guard Research and Development Center
1082 Shennecosett Road
Groton, CT 06340-2602

1 Ministry of Defence
Attn: Head, Naval Research
Dept. of Naval Architecture and Marine Engineering and Development
P.O. Box 20702
2500 ES, The Hague
The Netherlands

1 FOI Swedish Defence Research Agency
Attn: Eva Dalberg
Grindsjon Research Centre
SE-147 25 Tumba
Sweden

1 Department of Defence (Navy Office)
Attn: Director Naval Ship Design
Campbell Park Office CPI-505
Canberra ACT 2600
Australia

1 Document Exchange Centre
Defence Information Services Branch
Department of Defence
Campbell Park Offices CP2-5-08
Canberra ACT 2600 Australia

1 Bundesamt fur Wehrtechnik und Beschaffung - SG I 3
Postfach 7360
6057 Koblenz
Germany

1 Dr. John Duncan
Defence Equipment and Support
Abbey Wood
Mail Point 8014
BRISTOL BS34 8JH
UK

16 TOTAL LIST PART 2 (13 CDs, 3 Hard Copies)

35 TOTAL COPIES REQUIRED (30 CDs, 5 HARD COPIES)

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Martec Limited
Halifax, Nova Scotia

2a. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC JUNE 2010

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Development of a High Level Architecture Federation of Ship Replenishment at Sea: Final Report

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Bleichman, D.; Langlois, R.; Lichodzijewski, M.; Brennan, D.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
October 2011	234	11

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Atlantic
PO Box 1012, Dartmouth NS B2Y 3Z7, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11gv01	W7707-063647/001/HAL

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Atlantic CR 2011-261	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

In order to carry out the Navy's mission effectively, fleet units must be capable of remaining at sea for prolonged periods of time, possibly in areas of the world where friendly re-supply ports are not available, and remain fully ready to carry out any assigned tasks. Supply ships are equipped to replenish combatants underway with fuel, ammunition, provisions, and spare parts.

The Canadian Department of National Defence is looking to use a simulation infrastructure called the High Level Architecture (HLA) to provide integrated, joint simulation environments for the development of tactics and doctrine as well as for training and systems acquisition. In order to support this effort, the Warship Performance Section at DRDC Atlantic has initiated a research and development effort aimed at producing a simulation of ship replenishment at sea.

The objective of the proposed technical solution described in this report is to provide a simulation environment that models the interactions between the various components in order to simulate conditions that lead to the adverse events of payload immersion and RAS gear breakage.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

High Level Architecture; Replenishment at sea; Ship motions; Simulation

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca